AMSTRAD

GRAPHISME EN TROIS DIMENSIONS

THOMAS LACHAND-ROBERT



AMSTRAD
GRAPHISME EN TROIS DIMENSIONS

ISBN 2-7361-0157-4

THOMAS LACHAND-ROBERT

AMSTRAD
GRAPHISME EN TROIS DIMENSIONS



Paris • Berkeley • Düsseldorf

DANS LA MÊME COLLECTION

Amstrad jeux d'action, P. Monsaut
Amstrad premiers programmes, R. Zaks
Amstrad 56 programmes, S.R. Trost
Amstrad guide du BASIC et de l'AMSDOS, J.-L. Gréco/M. Laurent
Amstrad exploré, J. Braga
Amstrad programmation en assembleur, G. Fagot-Barraly
Amstrad guide du graphisme, J. Wynford
Amstrad CP/M 2.2, A. d'Hardancourt
Amstrad astrologie, numérologie, biorythmes, P. Bourgault
Amstrad Multiplan, Amstrad
Amstrad CP/M plus, A. d'Hardancourt
Amstrad Astrocalc, G. Blanc/P. Destrebecq
Amstrad gagnez aux courses, J.-C. Despoine
Amstrad créer de nouvelles instructions, J.-C. Despoine
Amstrad Locoscript, B. Le Dû
Amstrad Logo, A. d'Hardancourt *(à paraître)*
Amstrad mise au point des programmes BASIC, C. Vivier/J. Jacob *(à paraître)*
Amstrad programmes en langage machine, S. Webb *(à paraître)*
Amstrad guide du DOS, Amstrad *(à paraître)*
Amstrad introduction à la programmation en assembleur du Z80, A. d'Hardancourt *(à paraître)*
Amstrad systèmes d'exploitation, Amstrad *(à paraître)*
Amstrad programmes scientifiques, Y. Muggianu/M. Lamarche/P.-M. Beaufils *(à paraître)*
Amstrad routines en assembleur, J.-C. Despoine *(à paraître)*
Amstrad jeux en assembleur, E. Ravis *(à paraître)*
Amstrad mieux programmer en assembleur, T. Lachant-Robert *(à paraître)*

SOMMAIRE

PRÉFACE 9

I. INTRODUCTION 13

1. Les éléments graphiques des Amstrad 14
2. Éléments mathématiques des Amstrad 14
3. Autres instructions 16
4. Instructions supplémentaires des CPC 664 et 6128 17
5. Coup d'œil en avant 18

II. RAPPELS DE GÉOMÉTRIE ET DE MATHÉMATIQUES 21

1. Les coordonnées de l'espace et du plan 22
2. Éléments de trigonométrie 27
3. Les fonctions mathématiques de votre Amstrad 34
4. Les vecteurs, une chose bien commode 36
5. Application des vecteurs : équation d'un plan, d'une droite 40
6. Les transformations de l'espace : projections, similitudes 41
7. Applications affines et matrices 43

III. L'ART DU VOYEUR OU COMMENT REGARDER SOUS TOUS LES ANGLES 47

1. Tracer le repère sous un point de vue défini par deux angles en sphériques 48
2. Autres définitions du point de vue 50
3. Appliquer aux axes une transformation de l'espace 52
4. Animation 53
Réponses aux questions posées dans ce chapitre 56

IV. UN PLOT BALADEUR OU COMMENT TRACER N'IMPORTE QUOI DANS LE PLAN 59

1. Tracé sur un plan par un déplacement de plot 60
2. Enregistrement du tracé : point par point, sommet par sommet 63

3. Explication du Programme IV.1 : lever le crayon 65
4. Usage des couleurs 67

V. L'ART DE L'ILLUSION OU COMMENT REPRÉSENTER UN DESSIN SUR UN PLAN INCLINÉ 71

1. L'illusion de la perspective 72
2. Recto verso 75

VI. LES POLYÈDRES SOUS TOUS LES ANGLES 79

1. Qu'est-ce qu'un polyèdre ? 80
2. Définition du polyèdre : sommets, faces et arêtes.............. 82
3. Dessin et déplacement du polyèdre (utilisation par les faces) 87
4. Polyèdres convexes et concaves. Avantages, inconvénients 89
5. Polyèdres à répétition cylindrique 92
6. Polyèdre convexe d'une famille de points........................ 99
7. Polyèdres « étagés » 107
8. Suppression des parties cachées des polyèdres convexes 116
Réponses aux questions de ce chapitre 122

VII. TOUJOURS PLUS LOIN, TOUJOURS MIEUX 125

1. S'adapter aux particularités des objets tracés 126
2. Le labyrinthe ou l'appartement de rêve 126
3. Les faces bombées : surfaces polynomiales - Principe du tracé d'horizon - Bordures 140
4. Exemples 160

VIII. L'OMBRE ET LE FOND DES CHOSES : COUPONS, TRANCHONS, PLAQUONS.................................... 163

1. Ombres et reflets : les miracles de la projection 164
2. Principe d'une section. Application aux polyèdres 172
3. Principe d'une coupe. Application aux polyèdres............... 175
Réponses aux questions de ce chapitre 180

IX. LA NAPPE À CARREAUX : TRACÉ D'UNE SURFACE QUELCONQUE EN TROIS DIMENSIONS 181

1. Définition. Principes 182
2. Usage des coordonnées non cartésiennes. Usage des fonctions mathématiques 186
3. Cas essentiel : $z = f(x, y)$ 188
4. Une astuce spéciale : parties cachées sur de telles surfaces.... 190
5. Exemples et utilisation 195

X. LES PERSPECTIVES OPTIQUES 201

1. Pourquoi la perspective optique ? 202
2. Oui, mais comment ? Manière approchée 202
3. Oui, mais comment ? (suite). Manière scientifique 204
4. Exemple pratique : polyèdres 206

XI. POUR CONCLURE... 211

1. Optimisation des animations 212
2. Optimisation des calculs 212
3. Des inconvénients du BASIC : autres langages utilisables 213
4. Et dans l'avenir ? 214

ANNEXES :

A. THÉORIE ET PRATIQUE RÉSUMÉES DE LA DÉRIVATION 217

1. Notion de tangente à une courbe 218
2. Équation d'une tangente. Dérivée 220
3. Calcul d'une dérivée 220
4. Dérivée partielle 224
5. Utilisation 225

B. ÉQUATIONS IMPLICITES. RÉSOLUTION PRATIQUE D'ÉQUATIONS 227

1. Résolution d'une équation implicite : tracé de sa courbe représentative 228
2. Exemple d'application : surface $F(x, y, z) = 0$ 229
3. Résolution d'équation à une seule variable 230
4. Exemple d'application : dessin sur une surface 232

PRÉFACE

Ce livre tout entier part d'une constatation très simple, mais importante : le monde dans lequel nous vivons a trois dimensions, mais l'écran d'un ordinateur, tout comme une feuille de papier, n'en a que deux. D'où un évident problème, dès qu'il s'agit de représenter sur cet écran un objet tridimensionnel.

Et pourtant, cela n'est pas aussi difficile qu'on se l'imagine fréquemment. Des effets souvent superbes (voyez les figures de ce livre) peuvent être obtenus à partir de programmes qui ne dépassent pas une centaine de lignes, au grand maximum. C'est pourquoi l'on peut regretter que d'innombrables jeux, sur l'Amstrad comme sur les autres appareils, n'utilisent pas, ou mal, le graphisme en trois dimensions, là où ce serait parfois nettement plus esthétique. Seulement voilà, pour arriver à faire du graphisme en trois dimensions, il faut connaître un peu de mathématiques, et surtout beaucoup d'astuces !

C'est cette connaissance que cet ouvrage a la prétention de donner à toutes et à tous. Que vous soyez mathématicien, ou que vous ignoriez ce qu'est une racine cubique, peu importe. Le Chapitre II est là pour vous aider. Et pas seulement lui ; outre les dix-huit programmes de ce livre, quelque 45 schémas explicatifs assistent le texte pour vous en donner une compréhension optimale. Ces schémas ont été réalisés à la main : je vous prie donc d'avance de me pardonner d'éventuelles imperfections. Outre ces schémas, vous trouverez aussi dix-huit illustrations obtenues sur imprimante Amstrad (type DMP-1) grâce à un programme spécial. Au total, cela fait trente pages de figures environ. Ces figures sont numérotées en fonction des chapitres auxquels elles se rapportent, avec une lettre caractéristique (par exemple I.A., VII.C, etc.).

De même les programmes, tous en BASIC et aussi simples que possible, sont numérotés avec un chiffre arabe derrière le nombre romain de leur chapitre de rattachement : VI.3, VII.2, etc.

Je tiens à préciser immédiatement les points suivants :

- Primo, ces programmes ont tous été réalisés sur un Amstrad CPC 464. Bien évidemment, cela n'empêche pas les possesseurs d'autres machines, et plus particulièrement de matériel Amstrad, de les utiliser à leur profit. Ils sont tous compatibles avec les autres CPC. Pour les autres matériels, une transcription sera éventuellement nécessaire (voir le Chapitre I à ce sujet).

- Secundo, ils ont tous été longuement testés. S'ils ne marchent pas bien, vérifiez que vous les avez correctement entrés : une toute petite erreur peut suffire à les stopper complètement, chacun le sait.
- Tertio, j'ai jugé essentielle, avant toute chose, leur simplicité. Ils ne sont donc pas optimaux. D'ailleurs, je serai ravi de recevoir toute suggestion me permettant de les simplifier encore.
- Enfin, et surtout, ces programmes ne sont pas des buts en eux-mêmes : ils sont essentiellement destinés à être rajoutés à des programmes personnels, que ce soit de jeux ou semi-professionnels. Et ils ont surtout valeur d'exemple. Le but de cet ouvrage n'est pas en effet de vous enfermer dans le cadre rigoureux de programmes tout faits, mais bien au contraire de vous permettre de réaliser vous-même les programmes qui seront exactement adaptés à vos exigences. Il n'y a pas de limite à l'imagination, surtout quand elle peut courir sur trois dimensions !

Avant de vous laisser à votre lecture, je dois préciser quelques points. Tout d'abord, certains programmes, les plus longs, ou les plus difficiles, sont doublés d'un organigramme qui permettra au programmeur qui les connaît un peu de mieux s'y retrouver. Toutefois, pour des raisons de simplicité, les conventions de ces organigrammes ne sont pas strictement semblables à celles en usage, et de nombreuses expressions s'y trouvent abrégées. La liste de ces conventions et de ces abréviations figure en fin de préface.

D'autre part, pour que vous puissiez vous-même vérifier que vous avez bien compris certains points délicats et importants, certains chapitres sont dotés de questions dans le texte. La réponse à ces questions se trouve séparée en fin de chapitre. Est-il besoin de dire que je vous recommande d'essayer d'y répondre, quand vous les rencontrez dans le texte, sans vous précipiter vers la solution ? Si vous trouvez la solution facilement, bravo, sinon vérifiez que vous avez bien tout compris dans le paragraphe qui précède la question.

Enfin, j'ai jugé plus commode que les figures, programmes et organigrammes précèdent leurs commentaires dans l'ouvrage. De même, certains mots en italique, qui sont nouveaux, sont immédiatement suivis de leur définition. De même encore, chaque chapitre est précédé d'une courte introduction qui résume son contenu ; et le plan général de l'ouvrage est esquissé au Paragraphe 5 du Chapitre I.

L'ensemble se compose de onze chapitres, et de deux annexes pour ceux qui ont tout compris dans le Chapitre II. Alors il ne me reste plus... qu'à vous laisser les lire !

ORGANIGRAMMES

Notations et Symboles

Général
« Traitement »

~y : ~'z'
nbr? h

Entrée/Sortie
Print y: Print "z"
Input "nbr";h

Préparation

D a (s, 2)

Dimensionnement
Dim a(s, 2)

v f

Test vrai/faux

SOUSPROG

Vers sous-programme
Gosub...

Arrêt

Flag/Indicateur

Entrée
manuelle

t
a » b
$5

Opération
répétée
For t=a to b
step 5

Commentaire

ABRÉVIATIONS

z	≈ égale zéro
I **D**	incrémenté décrémenté
IO **SO**	<0 >0
F **R**	fin return
→	« devient » : **A→B** ≈ **B = A**
Σ	somme
c g s	change de signe
cœff.	cœfficients
coord	coordonnées
ds	dans
eq	équation
PB	plot baladeur
pl	plan
pt	point

I
INTRODUCTION

Ce court chapitre est destiné à récapituler les possibilités des diverses machines de la firme Amstrad et, par la même occasion, à faire le lien avec les autres machines. En effet, même si vous n'avez pas d'Amstrad, ce livre pourra vous être profitable, pour peu que vous connaissiez assez bien le BASIC de votre appareil pour pouvoir transcrire les programmes qui sont listés dans la suite.

Nous examinerons ensemble comment une telle transcription est possible, en regardant de plus près les éléments graphiques et mathématiques de l'Amstrad : à vous alors de faire (éventuellement) le tableau de correspondance des BASIC.

Enfin, nous esquisserons le plan global de cet ouvrage. Des figures, qui parsèment ce chapitre, vous permettront de mieux juger de ce qu'il est possible de faire avec un peu de réflexion et pas mal de travail informatique... ou avec le présent ouvrage en main.

1. LES ELEMENTS GRAPHIQUES DES AMSTRAD

Rappelons que les Amstrad (CPC 464, 664, 6128) ont, pour leur prix, une très bonne résolution : en mode 2 (haute résolution), 640 × 200 points, en mode 1 (moyen), 320 × 200, et en mode 0 (*coloré*), 320 × 100 points. Le nombre de couleurs évolue dans le sens inverse : deux, quatre ou seize suivant le mode, couleur du fond comprise.

Cet ouvrage traitant de questions graphiques assez fines, nous n'utiliserons jamais le mode 0.

Les Amstrad possèdent un jeu d'instructions graphiques très performant et de plus en plus, au fur et à mesure que l'on s'élève dans la gamme. Je rappelle que, pour que les programmes soient compatibles sur tous, seules les instructions du CPC 464 sont utilisées.

Comme tous les ordinateurs, l'Amstrad possède un curseur graphique de la taille d'un point. Ce curseur peut être déplacé sans être rendu visible avec l'instruction MOVE x, y, qui place le curseur à la position (x, y). L'instruction PLOT x, y, i rend le curseur visible en (x, y), dans la i^{ieme} couleur. L'instruction DRAW x, y, i trace un trait de l'ancienne position du curseur jusqu'à (x, y), dans l'encre i aussi. Toutes ces instructions sont doublées de leurs équivalentes *relatives* (MOVER, PLOTR, DRAWR), où (x, y) signifie : en se déplaçant de x horizontalement, et de y verticalement.

Les coordonnées absolues (x, y) sont mesurées par rapport à une origine, qui au début se trouve en bas à gauche sur l'écran, x étant compté positif vers la droite, et y vers le haut. L'origine peut être déplacée n'importe où sur ou hors de l'écran, grâce à l'instruction ORIGIN a, b, qui la place au point (a, b). Dans tous les cas, x et y peuvent varier de $-$ 32 760 à 32 760. Mais ils ne seront pas nécessairement sur l'écran. En effet, quel que soit le mode, l'écran est contenu dans le rectangle où x est compris entre 0 et 639, et y entre 0 et 399 (quand l'origine est placée en $(0, 0)$, bien entendu).

2. ELEMENTS MATHEMATIQUES DES AMSTRAD

De ce point de vue, les trois Amstrad ne diffèrent pratiquement pas.

L'Amstrad possède cinq opérateurs (+, −, *, /, ↑), sept fonctions *pures*

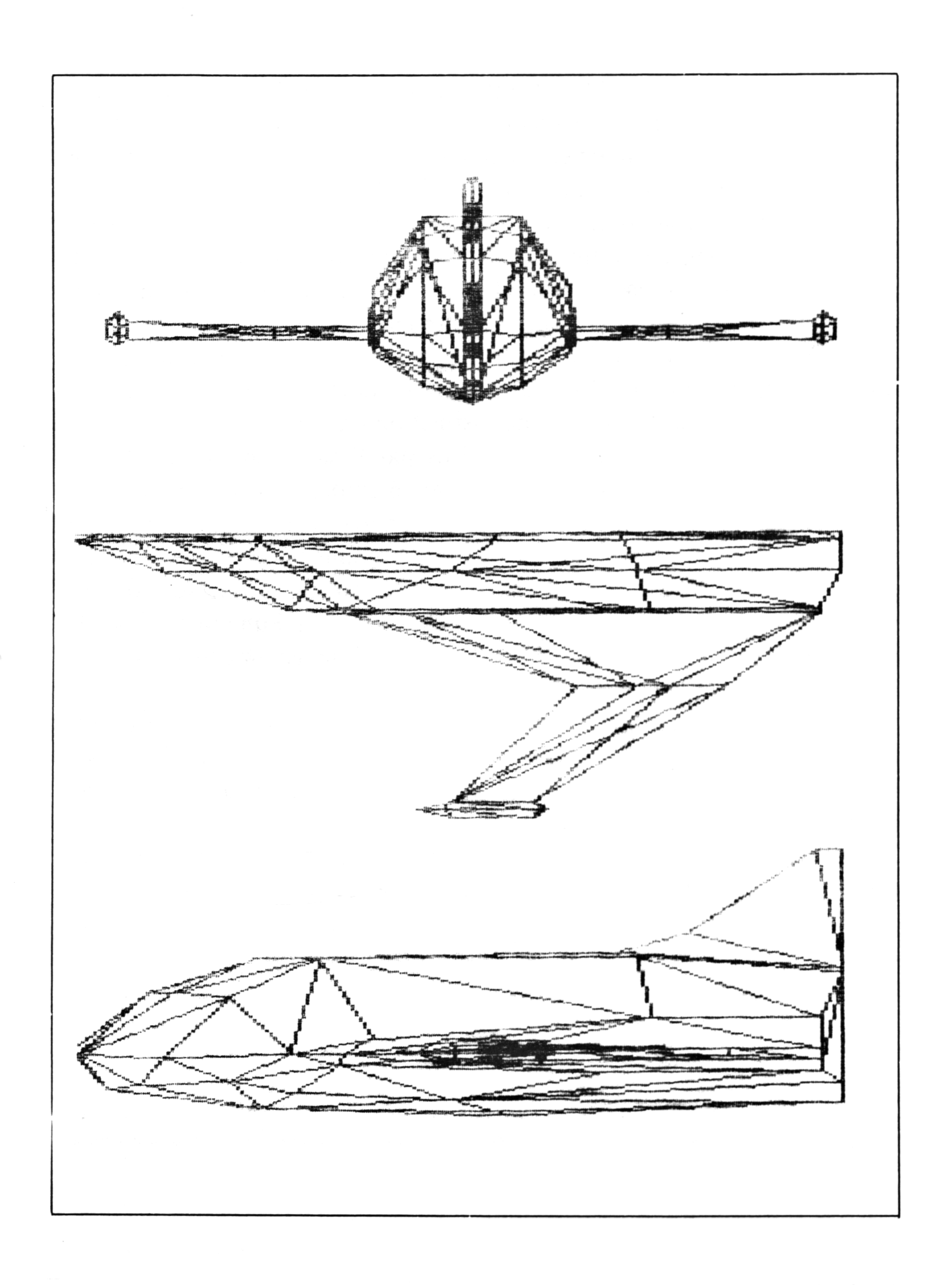


Figure I.A : Vaisseau spatial obtenu à l'aide du programme "Tous polyèdres" (Chapitre VI). Vue de face, vue de dessus (demie), vue latérale (toutes sans parties cachées).

(SIN, COS, TAN, ATN, EXP, LOG, LOG10) et deux autres (ABS, SGN), en plus de la racine carrée (SQR). En cela, il ne diffère guère des autres ordinateurs et machines à calculer, sauf de celles qui n'ont malencontreusement pas d'arc tangente (ATN).

Les fonctions trigonométriques notamment nous seront extrêmement utiles tout au long de ce livre (nous y reviendrons au chapitre suivant). Elles permettent déjà de suppléer à l'absence d'instruction *circle* permettant de faire des cercles (voir manuels à ce sujet).

Les Amstrad peuvent calculer sur des entiers (distingués par un signe % à la fin de leur nom, ou définis à l'avance comme entiers par l'instruction DEFINT) ou sur des réels (pas de distinction en général).

On peut y dimensionner des matrices par l'instruction DIM : DIM *a* (5, 6) dimensionne la matrice *a* à 5 lignes et 6 colonnes. C'est un ordre que nous utiliserons presque à chaque programme, car il est essentiel de pouvoir stocker des données quelque part. Certaines matrices serviront de pile, c'est-à-dire qu'on y empilera les données au fur et à mesure qu'elles viendront, et on les reprendra ensuite, comme on fait d'une pile d'assiettes qu'on place dans un placard. On peut aussi dimensionner des matrices d'entiers (en mettant un % derrière le nom, là aussi).

En principe, comme c'est normal, le calcul sur les entiers est plus rapide. Je n'ai toutefois pu mettre dans tous les programmes les DEFINT ou les % qui auraient été utiles. Je pense que le lecteur pourra le faire sans problème, le texte précisant ce qui est entier et ce qui ne l'est pas.

D'autre part, vous serez peut-être surpris de constater que dans tous ces programmes, l'ordre DEF FN, qui permet de définir une fonction, n'est jamais utilisé. De fait, je pense (sans en être certain, j'en conviens) que cet ordre ralentit les programmes, alors qu'il est pratiquement aussi simple de faire un GOSUB vers une routine qui calcule la valeur de la fonction.

3. AUTRES INSTRUCTIONS

Nous aurons à utiliser bien des fois l'ordre INKEY qui permet de tester si une touche du clavier est pressée, et si elle l'est en même temps que CTRL ou SHIFT (ou les deux). L'ordre INKEY$ est un peu différent : à tout moment, INKEY$ est égal au caractère tapé au clavier (s'il y en a un). Nous utiliserons souvent les deux lignes suivantes :

```
10 a$=INKEY$:IF a$="" GOTO 10
```

qui fait attendre la machine jusqu'à ce qu'une touche soit pressée, et :

```
WHILE INKEY(n)=-1 :WEND
```

qui place la machine dans une boucle conditionnelle WHILE-WEND tant que INKEY $(n) = -1$, c'est-à-dire tant que la touche n° n n'est pas pressée.

Nous utiliserons bien sûr constamment les ordres PRINT, INPUT, FOR... NEXT, GOTO, GOSUB, IF... THEN... ELSE, etc., qui n'appellent aucun commentaire, sauf pour PRINT. Rappelons que cet ordre peut être remplacé par un point d'interrogation (? a = PRINT a), et que l'instruction PRINT CHR$ (n) écrit le n^{ieme} caractère de la table des caractères. Les caractères usuels du clavier ont des numéros compris entre 32 et 122. Le caractère 13 est équivalent d'une pression de la touche ENTER. Le caractère 7 est un bip sonore. Dans tous les cas, le caractère peut être placé n'importe où sur l'écran par l'instruction LOCATE x,y, qui place le curseur de texte à la colonne x de texte, et à la ligne y (y est compris entre 1 et 25, et x entre 1 et 80 en mode 2, et 40 en mode 1). En outre, après l'exécution d'un ordre TAG, le curseur de texte se trouve au même endroit que le curseur graphique, ce qui permet de tracer un caractère vraiment n'importe où sur l'écran. Cet assujettissement est interrompu par l'ordre TAGOFF.

4. INSTRUCTIONS SUPPLEMENTAIRES DES CPC 664 ET 6128

Si vous possédez l'une de ces deux machines, vous pourrez facilement apporter quelques petites corrections aux programmes. En effet, ces appareils possèdent une douzaine d'ordres graphiques supplémentaires ou améliorés, ce qui vous permettra certainement des variations intéressantes.

En outre, un des ordres de texte supplémentaires, dont l'absence est assez gênante sur le 464, vous sera bien utile ; c'est l'ordre CLEAR INPUT qui vide la mémoire clavier. En effet, le BASIC mémorise les touches pressées durant l'exécution d'un programme, pendant un certain temps et jusqu'à concurrence de vingt. Dès lors, si vous appuyez sur les touches du

clavier ou sur la manette durant l'exécution, et c'est obligatoire pour de nombreux programmes, vous verrez au prochain INPUT, ou à la prochaine interruption, réapparaître tous ces caractères dont nul n'a que faire. Les heureux possesseurs des 664 et 6128 sauront donc, j'en suis persuadé, profiter de cette instruction que les propriétaires de 464 n'ont malheureusement pas...

5. COUP D'ŒIL EN AVANT

Qu'allons-nous voir à présent ? Je vais ici résumer très brièvement le contenu des autres chapitres.

Le Chapitre II est consacré à des notions de mathématiques élémentaires mais importantes : il vaut donc mieux le lire. Le Chapitre III est sans intérêt réel, mais il nous permettra de rentrer sans trop de "casse" dans l'univers tridimensionnel, d'abord toujours difficile. Le Chapitre IV nous permettra de nous reposer de cette courte aventure pour l'étude d'un petit truc du plan qui nous sera bien utile. Mais, au Chapitre V, le vif du sujet sera abordé, et l'ensemble deviendra vraiment très sérieux avec le Chapitre VI, consacré aux polyèdres, et qui est le plus long de tous. Le Chapitre VII vous grisera de l'air des sommets scientifiques : des questions vraiment complexes y seront abordées, et partiellement résolues.

Après un tel effort, le Chapitre VIII sera bien reposant, car il expose des principes très simples (qui sont d'ailleurs applicables à n'importe quoi). Le Chapitre IX réabordera une question importante (les surfaces), mais d'un point de vue tout à fait différent. Le Chapitre X vous ouvrira des perspectives nouvelles mais seulement superficiellement et de loin, car le reste est, je pense, déjà bien assez complexe, sans qu'on se mêle d'y apporter des détails qui sont tout de même, dans l'ensemble, assez raffinés. Le Chapitre XI constitue la conclusion de l'ensemble, comme le présent chapitre en est l'introduction.

Et maintenant que les présentations sont faites, il ne me reste plus qu'à vous souhaiter une bonne lecture.

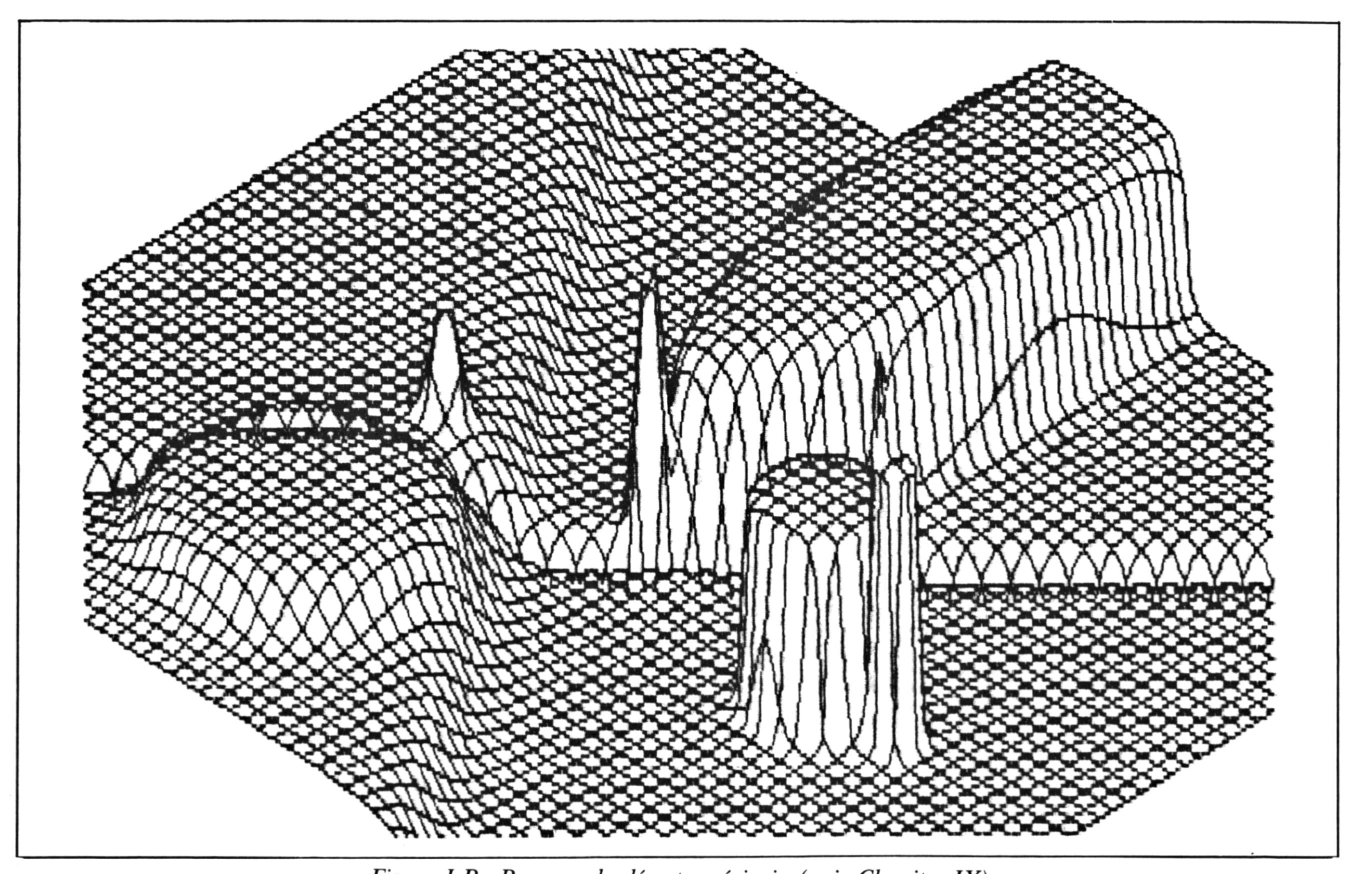


Figure I.B : Paysage de désert américain (voir Chapitre IX).

II

RAPPELS DE GEOMETRIE ET DE MATHEMATIQUES

Que vous soyez agrégé de mathématiques ou que vous n'ayez jamais entendu parler de sinus, vous pouvez faire du graphisme en trois dimensions. Mais un minimum de maths, et plus spécialement de géométrie, est absolument nécessaire. Il ne s'agit pas de notions difficiles, au contraire. Il ne faut donc pas craindre de les aborder : cela vous permettra de comprendre la suite. Évidemment, si vous refusez absolument d'approcher, même de loin, un univers qui vous semble hostile, vous pouvez toujours passer au chapitre suivant et appliquer directement les formules, sans chercher à les comprendre, mais ce serait dommage, d'autant plus que vous risquez de ne pas pouvoir adapter certains de ces programmes à vos propres besoins, ce qui est le but principal de cet ouvrage.

Donc, il vaut mieux que vous vous y plongiez, même avec effort : on n'a rien sans rien, n'est-ce pas ? Naturellement, rien ne vous empêche de *décrocher* en cours de route, certaines notions n'étant pas absolument nécessaires.

Si, par contre, vous êtes déjà très fort en sciences, ce chapitre peut (éventuellement) servir à vous rafraîchir un peu la mémoire, ou simplement à préciser mes notations. Mais il est fait pour des personnes qui ne savent pratiquement rien en géométrie : certains points vous paraîtront alors peut-être trop détaillés. Je m'en excuse d'avance, mais il faut bien satisfaire tout le monde...

A présent, si vous savez seulement ce que c'est qu'un point, une droite et un plan, laissez-vous mener ; dans quelques pages, vous saurez le reste.

1. LES COORDONNEES DE L'ESPACE ET DU PLAN

On a souvent besoin de préciser sa position. Il existe pour cela plusieurs moyens. Par exemple, indiquer le numéro d'une borne kilométrique, sur une route ; ou sa latitude et sa longitude ; etc.

Vous pouvez avoir besoin de vous repérer sur une droite ou une courbe, sur un plan ou une surface, dans l'espace en général.

Sur une droite c'est très facile, il suffit de choisir un point arbitraire sur la droite (le centre d'une grande ville, pour une route par exemple), et de donner votre distance à ce point, que nous appellerons *origine,* ou O : c'est ce que fait la borne kilométrique.

Sur un plan, le sol, ou votre plancher par exemple, c'est un peu plus subtil. Un moyen simple consiste à donner, sur le plancher, la distance du point où vous vous trouvez par rapport à deux des murs. L'intersection de ces murs sera l'origine. Les murs en eux-mêmes coupent le plancher selon des droites. Ces droites, nous les appellerons axes.

Plan

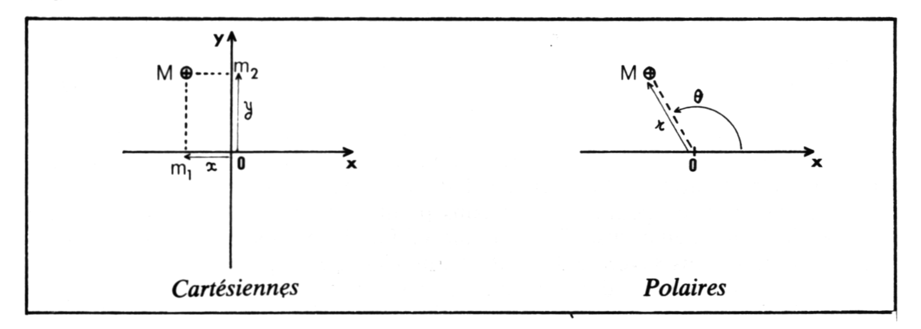


Figure II.A.a

Espace

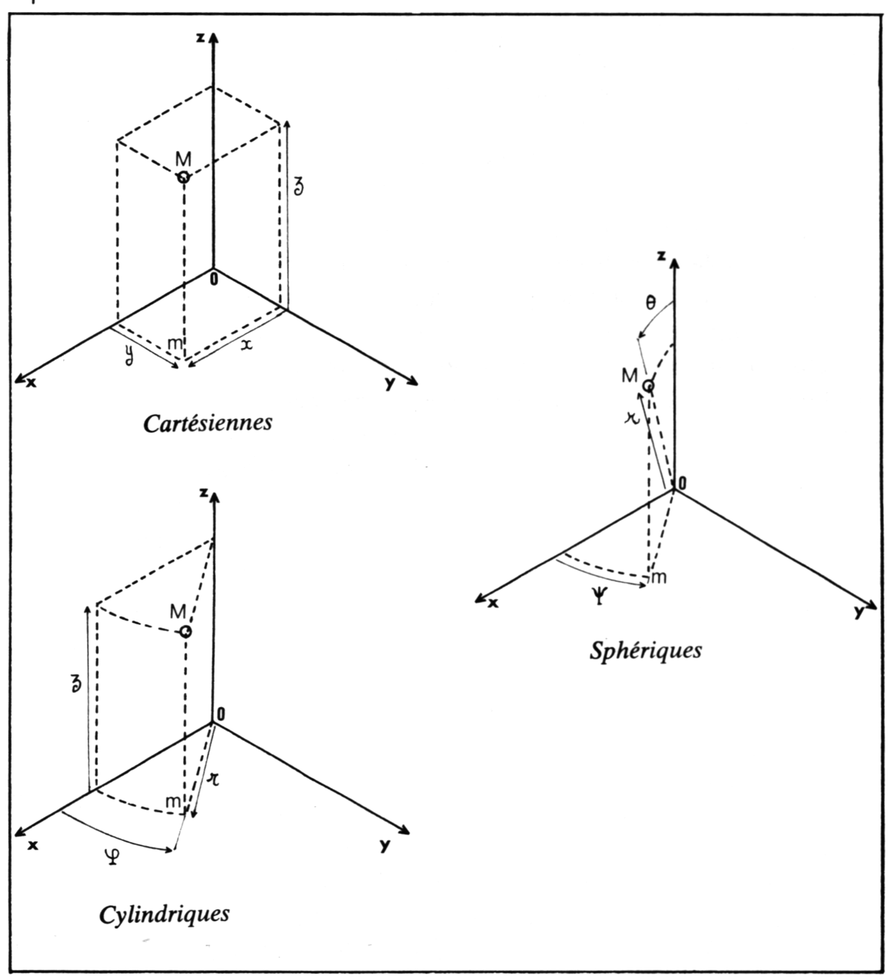


Figure II.A.b

Vu d'en haut, votre plancher sera donc à peu près semblable à ce que montre le schéma à gauche de la Figure II.A.*a* (p. 22) : vous reconnaissez l'origine O et les deux axes que nous appellerons Ox et Oy. Remarquez que ces deux *axes* sont ainsi nommés parce qu'en plus d'être des droites, ils sont dotés de petites flèches à l'une de leurs extrémités ; ce sont des droites *orientées*.

A présent, comment repérer le point M, placé quelque part sur ce plan ? C'est tout simple. M a deux projections m_1 et m_2 respectivement sur Ox et Oy, obtenues en abaissant des perpendiculaires (voir Figure II.A.*a*). Si l'on appelle x la distance de O à m_1, et y la distance de O à m_2, le couple (x, y) s'appelle *coordonnées cartésiennes* du point M. Tout point a ainsi deux coordonnées cartésiennes dans un plan, et inversement, si l'on se donne deux nombres, on peut trouver le point dont ils sont les coordonnées.

Remarquez une chose importante : ces deux nombrex x et y ont un signe. Ce signe est plus (positif) si la flèche joignant O au point m concerné est dans le même sens que l'axe (comme pour y), et négatif sinon. Ainsi, dans le cas de figure, x est négatif, car la flèche joignant O à m_1 est dirigée dans le sens contraire de l'axe Ox. Cette distinction est importante, sans quoi il y aurait quatre points dans le plan pour chaque couple de coordonnées, et ce ne serait pas commode !

Les *coordonnées polaires* sont un autre moyen de repérer un point dans le plan. Elles utilisent les angles. Chacun sait, je pense, ce qu'est un angle. Nous verrons tout à l'heure comment le mesurer. Je prendrai l'habitude, pour distinguer les angles des distances comme x et y, d'utiliser pour les désigner des lettres grecques[(1)], ou de mettre un accent circonflexe dessus comme ceci : $\hat{a}$. Les coordonnées polaires, c'est une distance, notée r, et un angle θ. La première est la distance du point M à l'origine O : c'est ce que mesurent les joueurs de boule, en cas de litige. L'angle θ est l'angle séparant Ox de OM, avec son signe que nous verrons au paragraphe suivant. Tout cela est visible sur la Figure II.A.*a*, à droite.

Et pour l'espace ? Là, trois moyens sont possibles (il en existe d'autres encore, je donne les principaux).

Et tout d'abord les coordonnées cartésiennes, encore elles. Mais cette fois, il y en a trois : x, y, z. Regardez la figure II.A.*b*. Du point M à repérer, on abaisse une verticale (parallèle à Oz), et l'on obtient un point m dans le plan qui contient les deux axes Ox et Oy, plan que nous appellerons désormais xOy. Ce point m a, dans ce plan, deux coordonnées

(1) En particulier θ (thêta), ψ (psi), φ.(phi), α (alpha).

cartésiennes x et y, comme nous l'avons vu. En ajoutant donc à ce plan un troisième axe Oz qui lui est perpendiculaire, et en mesurant la distance entre m et M (positive si la flèche est dans le même sens que Oz, négative sinon), on obtient la valeur de la troisième coordonnée z. Voilà quelles sont les trois coordonnées cartésiennes (x, y, z) d'un point. Retenez-les bien, car ce sera surtout ce système qui sera utilisé dans la suite, les deux autres ne servant que dans certains cas.

Avant de passer aux sphériques, il existe un système mixte, dit à *coordonnées cylindriques*. Dans ce système, on repère m par ses coordonnées polaires dans le plan, r et φ (φ au lieu de θ pour éviter la confusion avec les sphériques). La signification de z reste inchangée.

Les *sphériques* sont l'homologue des polaires du plan. Mais ici, outre r qui est la distance de M à O encore, il y a deux angles. L'angle θ est l'angle séparant l'axe Oz de OM. L'angle ψ est l'angle séparant l'axe Ox de Om. L'axe Oy, ici, ne sert à rien, tout comme pour les cylindriques et les polaires.

Tout cela vous paraît peut-être très artificiel. C'est un tort. Un pilote d'avion, en indiquant son altitude et le point à la verticale duquel il se trouve, ne fait jamais que donner sa coordonnée z et le point m (qui peut être repéré en cartésiennes, mais aussi en cylindriques, par exemple : “Je suis à 300 pieds à la verticale du point situé à vingt kilomètres au sud-est de l'aéroport.” En effet, sud-est équivaut à + 45 degrés par rapport au sud). Le marin, lui, utilise les sphériques sans le savoir. Lorsqu'il précise sa latitude et sa longitude, il ne fait jamais que donner des angles qui sont des coordonnées sphériques ; cela est d'ailleurs très logique, car la Terre, et par conséquent la surface océane, est sphérique (enfin, à peu près). Évidemment r n'est pas alors précisé, car c'est une constante : la distance au centre de la planète, c'est-à-dire son rayon ; inutile de parler d'une chose qui ne change pas.

Vous voyez donc que tous les systèmes de coordonnées peuvent servir. Cependant les cartésiennes restent sans conteste les plus commodes, et nous les utiliserons presque toujours.

Je voudrais placer ici une remarque que vous avez peut-être déjà faite. Pour une droite il faut, pour se repérer, une coordonnée. Sur un plan, il en faut deux. Dans l'espace, il en faut trois. Ce nombre de coordonnées nécessaires pour se repérer dans un domaine donné est appelé *dimension* du domaine. Nous dirons ainsi que la droite est un domaine à une dimension, que le plan est bidimensionnel, et que l'espace (usuel) est, devinez... à trois dimensions.

Nous verrons au Chapitre IX que toutes les surfaces, et pas seulement les plans, sont bidimensionnelles. On s'y repère à l'aide de deux coordonnées appelées paramètres. Nous l'avons déjà vu : sur une sphère (la Terre, au niveau des eaux), on se repère à l'aide de deux paramètres, la latitude et la longitude, qui ne sont d'ailleurs que les deux dernières coordonnées sphériques. Vous voyez ainsi que certaines coordonnées sont surtout adaptées à certaines surfaces, ou à certaines courbes. On ne les choisit donc pas au hasard.

Cette dissertation philosophique sur la dimension étant achevée, je vais à présent donner, pour éviter de fâcheux mélanges, les formules qui permettent de passer d'un système à l'autre, car cela sert fréquemment. Si vous ne connaissez rien des sinus et consinus, passez directement au paragraphe suivant, puis revenez ici. Dans ces formules, comme dans toutes les autres, le point ($\cdot$) signifie multiplié par, cos est le cosinus, sin le sinus, etc., r^2 signifie r au carré (c'est-à-dire $r \cdot r$), sqr (r) la racine de r, et π le nombre pi.

FORMULES DE CHANGEMENT DE COORDONNÉES

PLAN

des cartésiennes aux polaires :

$$r = \text{sqr}\,(x^2 + y^2)\,; \quad \theta = \text{atn}\left(\frac{y}{x}\right) + (1 - \text{sgn}\,(x)) \cdot \frac{\pi}{2}\ ^{(*)}$$

des polaires aux cartésiennes :

$$x = r \cdot \cos(\theta)\,; \quad y = r \cdot \sin(\theta)$$

ESPACE

des cartésiennes aux cylindriques, des cylindriques aux cartésiennes :

comme dans le plan, en remplaçant θ par φ, et avec $z = z$.

des cartésiennes aux sphériques :

$$r = \text{sqr}\,(x^2 + y^2 + z^2)\,; \quad \theta = \text{atn}\left(\frac{\text{sqr}\,(x^2 + y^2)}{z}\right)$$

$$\psi = \operatorname{atn}\left(\frac{y}{x}\right) + (1 - \operatorname{sgn}(x)) \cdot \frac{\pi}{2} \quad (*)$$

des sphériques aux cartésiennes :

$$x = r \cdot \sin(\theta) \cdot \cos(\psi) \quad y = r \cdot \sin(\theta) \cdot \sin(\psi) \quad z = r \cdot \cos(\theta)$$

des sphériques aux cylindriques :

$$r = r \cdot \sin(\theta)\,; \quad z = r \cdot \cos(\theta)\,; \quad \varphi = \psi$$

(Le premier r est celui des cylindriques, les deux autres sont des sphériques.)

des cylindriques aux sphériques :

$$r = \operatorname{sqr}(r^2 + z^2)\,; \quad \theta = \operatorname{atn}\left(\frac{z}{r}\right); \quad \psi = \varphi \quad \text{(Remarque inverse.)}$$

(*) Sauf si $x = 0$; dans ce cas, l'angle vaut $\operatorname{sgn}(y) \cdot \frac{\pi}{2}$.

2. ELEMENTS DE TRIGONOMETRIE

Je dois avouer que j'ai toujours été surpris de voir l'effroi que suscite fréquemment chez les gens le mot de trigonométrie, comme si elle était le fondement même des mathématiques qui, hélas, demeurent de nos jours, malgré les efforts des enseignants, pour la plupart de nos concitoyens totalement absconses.

Je me fais fort de prouver en deux pages que la trigonométrie ne mérite pas cette terrible réputation, du moins dans ses fondements.

Tout part des angles. Tout le monde sait, intuitivement, en fait, ce qu'est un angle. Vous savez peut-être aussi qu'il existe en France deux moyens concurrents de les mesurer. De ces deux moyens, qui sont disponibles sur l'Amstrad, l'un est totalement artificiel, mais remonte à la nuit des temps ; le second est plus naturel, mais plus récent et pas du tout “dans les mœurs”.

Vous avez certainement reconnu les degrés et les radians. Pour les premiers, on décide (arbitrairement) que le plus grand angle possible, c'est-à-dire le tour entier sur soi-même, fait 360°. Le reste en résulte. L'angle que vous faites avec votre ancienne position en tournant sur vous-même d'un quart de tour par exemple (angle droit), fait 360/4 soit 90°. Cela, presque tout le monde le sait.

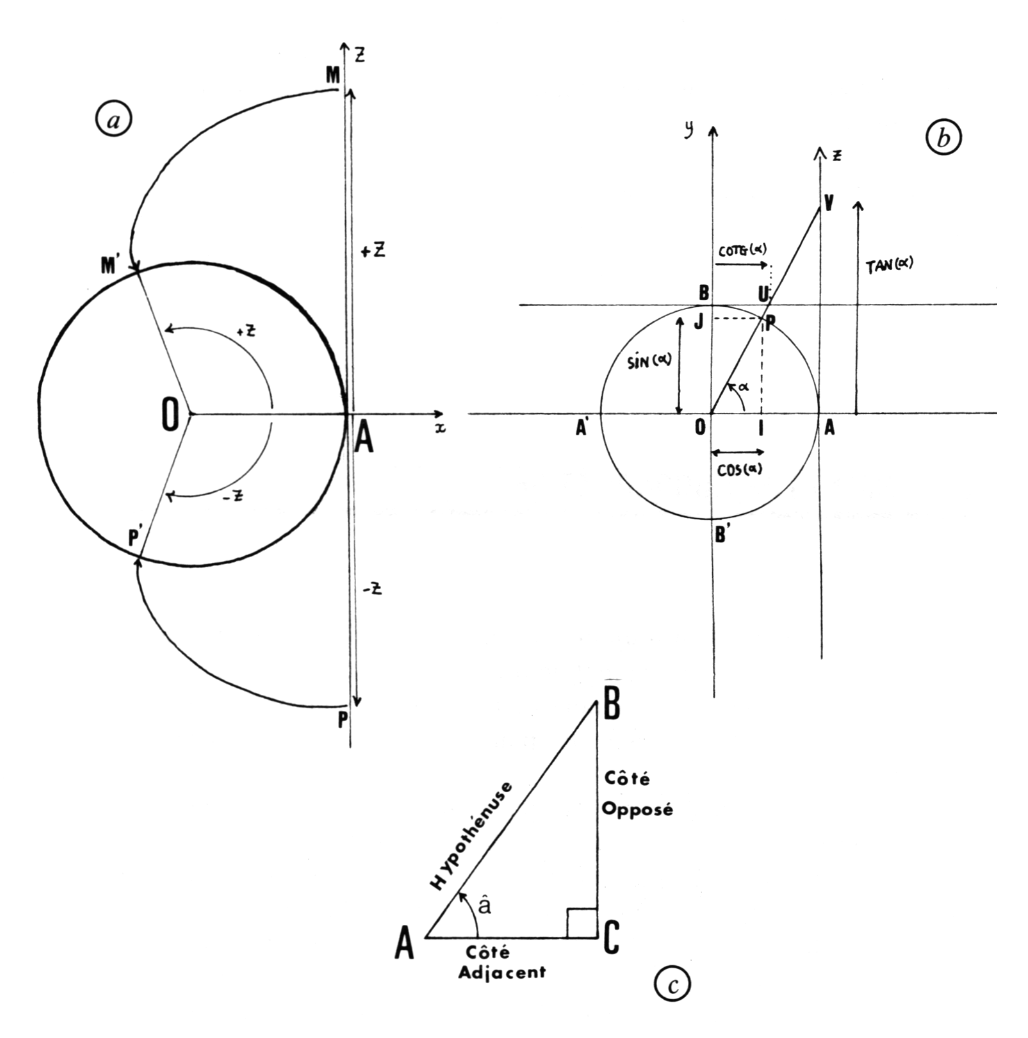


Figure II.B, schémas a, b, c.

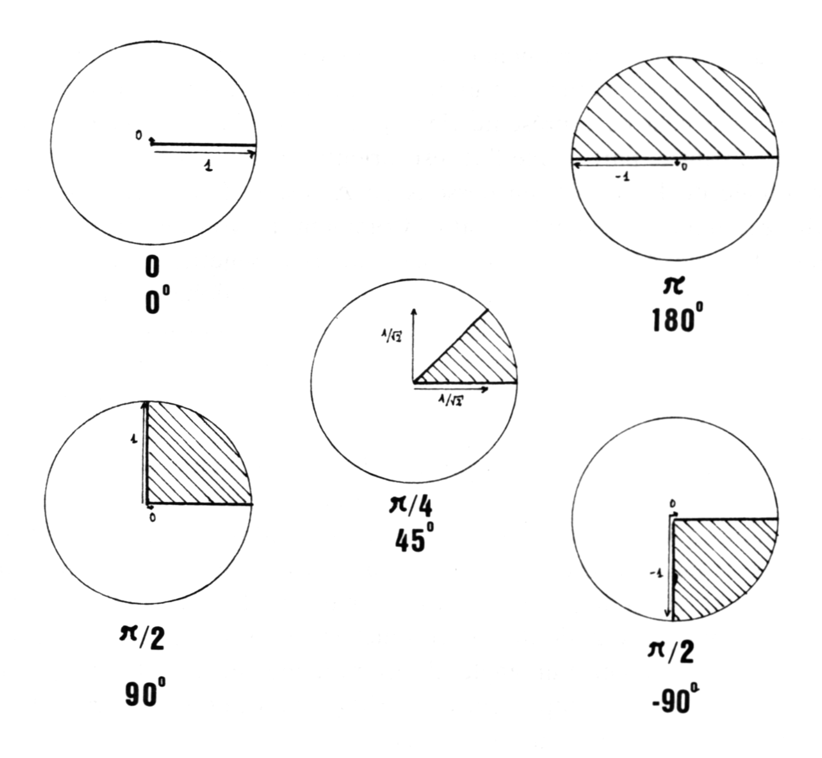


Figure II.B, schéma d.

Le second moyen de mesure, en radians, est expliqué par le schéma *a* de la Figure II.B (les lettres qui distinguent les schémas sont blanches, et entourées d'ellipses ou de cercles). Regardez bien le cercle de centre O qui y figure. A sa droite, on a mis, tout contre, un axe AZ, A étant le point de contact. A présent, sur cet axe, prenons un point M donné, de coordonnée Z (rappelez-vous : une seule coordonnée sur une droite). Imaginez ensuite que l'on remplace le segment AM par un bout de ficelle. Puis on enroule ce bout de ficelle (dont la longueur est Z) autour du cercle, comme autour d'un yoyo, après l'avoir attaché en A. L'autre bout finit par arriver sur le cercle en M′. Eh bien, l'angle séparant l'axe O*x* (ou OA, si vous préférez) de OM′ mesure exactement Z radians. Voilà, c'est tout simple, en fait.

Précisons simplement deux choses. D'abord, cela ne doit pas dépendre de l'unité de mesure prise sur l'axe, ni du rayon du cercle. On décide donc

que ce rayon vaut 1 (en mathématiques, on n'exprime pas les longueurs en mètres ou centimètres mais en unités, ce qui peut représenter n'importe quoi puisque cela ne représente rien, justement). La distance OA vaut donc 1. D'autre part, l'axe AZ est orienté (vers le haut). Ainsi, la coordonnée de P, qui est en dessous de A, est $-$Z négative). L'angle AOP′ est donc également négatif. Comment le voit-on ? C'est assez simple. L'angle AOM′ est dans le sens dit trigonométrique, c'est-à-dire dans le sens inverse des aiguilles d'une montre : il est donc positif. L'autre est dans le sens inverse (voyez la flèche) : il est donc négatif.

Remarquons enfin et surtout, primo, que le tour d'un cercle de rayon 1 vaut 2π (dites deux pi ; la lettre π représente le nombre 3,141 592...). Donc, 2π radians = 360°.

Secundo, une remarque qui vaut pour les deux systèmes de mesure : si vous tournez sur vous-même de, par exemple, 30°, puis que vous faites un tour complet (360°), vous aurez tourné de 390°. Mais le tour complet ou rien, c'est la même chose : il ne change pas votre position. Donc entre un angle de 390° et un angle de 30°, il n'y a pas de différence. D'une façon générale, vous pouvez ajouter ou retrancher 360° (ou 2π rad) autant de fois que vous le voulez à un angle, il restera le même. Dans la suite, nous prendrons des angles compris entre $-$ 180° ($-\pi$ rad) et + 180° (π rad), ce qui représente tous les angles possibles.

A partir de ces angles, nous pouvons trouver leur sinus, leur cosinus et leur tangente assez facilement.

Regardez attentivement le schéma *b* de la Figure II.B. On y voit de nouveau O, AZ, et le cercle de rayon 1, qui porte le nom de *cercle trigonométrique*. Un point P se trouve sur ce cercle, et on appelle α l'angle qui sépare OA de OP. Or, voyez que le point P peut être projeté sur les deux axes Ox et Oy, respectivement en I et J. Les mesures de OI et OJ sont en fait les coordonnées de P.

Par définition, on appelle *cosinus de l'angle* α (cos (α)) la première coordonnée de P, soit OI, et *sinus de l'angle* α (sin (α)) la seconde, soit OJ. C'est tout. Mais c'est bien plus qu'il ne paraît. Nous le verrons tout à l'heure.

Regardez encore la figure. La droite OP coupe l'axe BT en U, et l'axe AZ en V. Toujours par définition, on appelle *tangente de l'angle* α (tg (α) ou tan (α)) la coordonnée Z de V (soit AV), et *cotangente de l'angle* α (cotg (α) ou cotan (α)) la coordonnée T de U, soit BU.

Ces quatre valeurs ont bien évidemment un signe, positif dans le cas de figure.

De cette figure, on peut déduire des tas de propriétés très simples, mais essentielles. Par exemple, comme AO = OB = 1 (cercle de rayon 1), que OA′ = OB′ = − 1 (orientation contraire), et que I est forcément situé entre A′ et A, et J entre B′ et B, on en déduit que le sinus et le cosinus sont forcément compris entre − 1 et 1. D'autre part, pour le triangle rectangle OIP et suivant le théorème de Pythagore :

$$OP^2 = OI^2 + PI^2.$$

Or $OP = 1, OI = \cos(\alpha), PI = OJ = \sin(\alpha)$, donc :

$$\boxed{\cos^2(\alpha) + \sin^2(\alpha) = 1}$$

Cette relation capitale est vraie pour tout angle.

Suivant le théorème de Thalès, vu que IP est parallèle à AV, et que OIA d'une part et OPV d'autre part sont alignés, alors :

$$\frac{AV}{OA} = \frac{IP}{OI}$$

Or OA = 1, donc :

$$\boxed{\tan(\alpha) = \frac{\sin(\alpha)}{\cos(\alpha)}}$$

De la même manière :

$$\text{cotan}(\alpha) = \frac{\cos(\alpha)}{\sin(\alpha)}$$

donc

$$\boxed{\text{cotan}(\alpha) = \frac{1}{\tan(\alpha)}}$$

Toutes ces relations sont vraies pour tout angle. Cela explique que votre Amstrad n'ait pas de fonction cotangente, car il suffit d'inverser la tangente pour l'avoir. C'est aussi pourquoi on utilise peu la cotangente : désormais, nous n'en parlerons presque plus.

Voilà l'essentiel de ce qu'il faut savoir en trigonométrie. Mais à quoi tout cela sert-il ? Regardez le schéma *c* de la Figure II.B. On y voit un triangle

rectangle en C, c'est-à-dire que l'angle en C est droit, ce qu'on note en y mettant un petit carré.

Soit $\hat{a}$ l'angle en A. Les trois côtés du triangle seront appelés hypoténuse pour le plus long (AB), côté opposé à l'angle (BC), et côté adjacent à l'angle (AC). Pour connaître les valeurs de ces différents côtés, les uns en fonction des autres, connaissant $\hat{a}$, on dispose des trois formules suivantes :

$$\sin(\hat{a}) = \frac{BC}{AB}; \ \cos(\hat{a}) = \frac{AC}{AB}; \ \tan(\hat{a}) = \frac{BC}{AC}$$

ce qui se résume dans le mot mnémotechnique SOHCAHTOA, c'est-à-dire : Sinus égale côté Opposé sur Hypoténuse, Cosinus égale côté Adjacent sur Hypoténuse, Tangente égale côté Opposé sur côté Adjacent. Ces trois formules sont absolument fondamentales en géométrie.

Notons qu'on les connaît déjà dans le cas du triangle OIP, car l'hypoténuse OP valant 1, on retrouve :

$$\sin(\alpha) = IP\,(= OJ)\,;\ \cos(\alpha) = OI\,;\ \tan(\alpha) = \frac{IP}{OI} = \frac{\sin(\alpha)}{\cos(\alpha)}$$

C'est à cause de ces formules dites du triangle que les sinus, etc., sont essentiels. Cela justifie le terme de trigonométrie qui signifie mesure du triangle. En effet, en géométrie, il y a des triangles partout.

Pour mieux connaître ces fonctions trigonométriques, le schéma *d* de la Figure II.B vous en donne la valeur pour quelques angles :

En $0\,(0°)$: $\sin(0) = 0\,;\ \cos(0) = 1\,;\ \tan(0) = 0$

En $\frac{\pi}{2}\,(90°)$: $\sin\left(\frac{\pi}{2}\right) = 1\,;\ \cos\left(\frac{\pi}{2}\right) = 0\,;$

pas de tangente (division par zéro).

En $\pi\,(180°)$: $\sin(\pi) = 0\,;\ \cos(\pi) = -1\,;\ \tan(\pi) = 0$

En $-\frac{\pi}{2}\,(-90°$, ou $+270°)$: $\sin\left(-\frac{\pi}{2}\right) = -1\,;\ \cos\left(-\frac{\pi}{2}\right) = 0\,;$

pas de tangente.

Enfin au demi-angle droit,

$\frac{\pi}{4}$ (45°), où sin et cos sont égaux :

$$\sin\left(\frac{\pi}{4}\right) = \cos\left(\frac{\pi}{4}\right) = \frac{1}{\text{sqr }(2)}\,;\ \tan\left(\frac{\pi}{4}\right) = 1$$

Pour trouver commodément la valeur de ces fonctions en certains angles, on dispose de formules très utiles que je donne ici (valables pour tout x et tout y) :

FORMULES DE TRIGONOMÉTRIE

$-x$? $\sin(-x) = -\sin(x)\,;\ \cos(-x) = \cos(x)\,;$
$\tan(-x) = -\tan(x)$

$x+\pi$? $\sin(x+\pi) = -\sin(x)\,;\ \cos(x+\pi) = -\cos(x)\,;$
$\tan(x+\pi) = \tan(x)$

$\frac{\pi}{2}-x$? $\sin\left(\frac{\pi}{2}-x\right) = \cos(x)\,;\ \cos\left(\frac{\pi}{2}-x\right) = \sin(x)\,;$

$$\tan\left(\frac{\pi}{2}-x\right) = \frac{1}{\tan(x)}$$

$2x$? $\sin(2x) = 2\cdot\sin(x)\cdot\cos(x)$
$\cos(2x) = \cos^2(x) - \sin^2(x) = 2\cdot\cos^2(x) - 1 = 1 - 2\cdot\sin^2(x)$

$x+y$? $\sin(x+y) = \sin(x)\cdot\cos(y) + \sin(y)\cdot\cos(x)$
$\cos(x+y) = \cos(x)\cdot\cos(y) - \sin(x)\cdot\sin(y)$

$x-y$? $\sin(x-y) = \sin(x)\cdot\cos(y) - \sin(y)\cdot\cos(x)$
$\cos(x-y) = \cos(x)\cdot\cos(y) + \sin(x)\cdot\sin(y)$

Ces formules nous serviront assez fréquemment dans la suite. N'hésitez pas à vous y reporter si vous avez un doute.

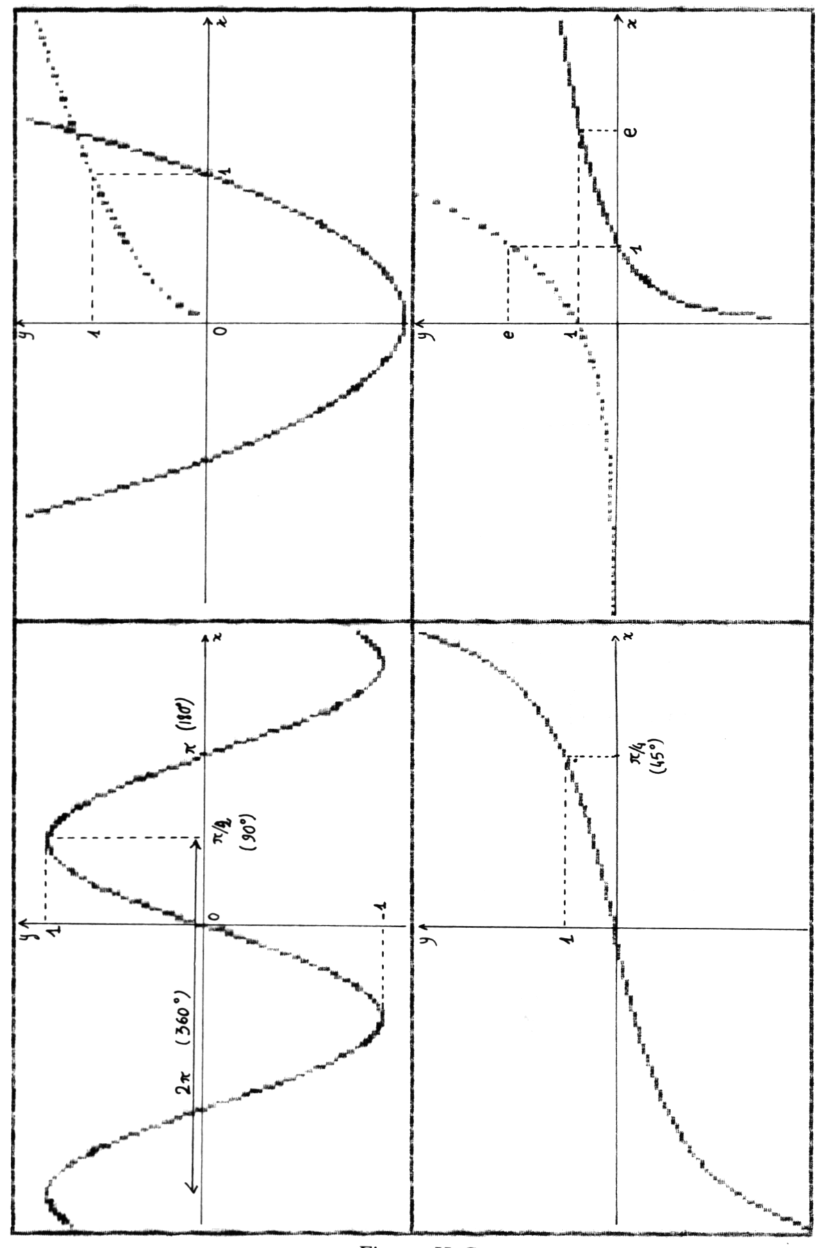


Figure II.C

Outre les trois fonctions trigonométriques déjà vues, l'Amstrad possède quatre fonctions que nous utiliserons de temps à autre, et sur lesquelles je souhaite dire juste quelques mots : il s'agit de la racine carrée sqr, de l'arc tangente atn, de l'exponentielle exp et du logarithme log.

La Figure II.C représente quelques fonctions de l'Amstrad. En effet on peut représenter une fonction graphique par y = fonction de x, en faisant varier x.

Ainsi, en haut et à gauche, on a $y = \sin(x)$. La courbe obtenue porte le nom de sinusoïde. On reconnaît quelques valeurs du sinus (voir ci-dessus). On vérifie que le sinus oscille entre -1 et 1. Enfin, on remarque que la courbe se répète tous les 2π radians, ce qui n'a rien de surprenant puisque les angles eux-mêmes se répètent. On dit que cette courbe est *périodique.* Notez que la courbe du cosinus est exactement la même, mais décalée de $\pi/2$ vers la droite, de sorte que le maximum, le sommet, se trouve sur l'axe des y.

Juste en dessous, la courbe $y = \tan(x)$. Cette courbe est infinie vers le haut et le bas, lorsque x s'approche de $\pm\ \pi/2$ (rappelez-vous que la tangente n'existe pas pour ces valeurs, elle serait infinie). Là encore, la courbe est périodique, mais de période π (voir $\tan(x + \pi) = \tan(x)$).
En haut à droite, la courbe en traits pleins est la courbe $y = x^2$.
Elle porte le nom de *parabole*. En pointillé, la courbe y'= sqr (x). C'est aussi une parabole, mais couchée, et il en manque la moitié. Ces deux fonctions sont dites réciproques, en ce sens que si y = sqr (x), alors $x = y^2$, et inversement si $y = x^2$, alors x = sqr (y)[1]. Notons que la courbe en pointillé se trouve entièrement à droite de l'axe Oy : la racine carrée d'un nombre négatif n'existe pas.

La courbe représentant la fonction atn n'est pas représentée. Atn (arc tangente) est la réciproque de la tangente : si y = atn (x), alors $\tan(y) = x$; ou encore, si vous préférez, $x = \tan(\text{atn}(x))$. Votre Amstrad donne l'atn entre $-\pi/2$ et $\pi/2$. N'oubliez pas, en effet, que

$$\tan(y) = \tan(y + \pi) = \tan(y + 2\pi) = \ldots$$

et donc que l'arc tangente est déterminée à un certain nombre de fois π près (on dit : “modulo π”).

(1) Avec x et y positifs.

Le dernier schéma, en bas à droite de la figure, représente les fonctions exp et log. La première, en traits pleins, a une valeur facile à comprendre ; pour tout x, $\exp(x) = e^x$ (e à la puissance x), où $e = \exp(1)$ est un certain nombre égal à 2,71 828 183... dont l'origine serait trop compliquée à expliquer ici. Le logarithme dit népérien (du nom du mathématicien Neper), ou naturel, est la réciproque de l'exponentielle :

$$y = \log(x) \quad \text{si} \quad x = \exp(y) = e^y$$

Par exemple, $1 = \exp(0)$ et $e = \exp(1)$, donc $\log(1) = 0$ et $\log(e) = 1$.

Vous pouvez encore vérifier cela en prenant un nombre au hasard, par exemple 3. Tapez sur votre ordinateur : $e = \exp(1)$. Puis : PRINT exp (3). Vous obtenez 20,085... Puis : PRINT $e \uparrow 3$. Vous obtenez le même nombre. A présent, tapez : PRINT log (20,085...) (en recopiant le nombre). Vous obtenez 3. C.Q.F.D.

L'intérêt pratique du logarithme sort du cadre de cet exposé. Celui de l'exponentielle est très variable. Il permet par exemple de calculer $x^y = \exp(y \cdot \log(x))$.

Notons que la fonction 10^x est semblable à exp (qui est e^x), en remplaçant e par 10. Elle a aussi une réciproque, le logarithme décimal, que votre Amstrad note log 10 : $\log 10(x) = y$ équivaut à $x = 10^y$.

Cet aperçu des fonctions mathématiques de votre ordinateur étant fait, nous allons passer à un sujet tout différent, en nous replongeant au cœur de la géométrie.

4. LES VECTEURS, UNE CHOSE BIEN COMMODE

Nous utiliserons beaucoup les vecteurs dans ce livre. Non qu'ils soient absolument essentiels, mais ils sont si commodes que ce serait dommage de s'en priver.

Pour expliquer de quoi il s'agit, je dois d'abord dire ce qu'est un bipoint. Un *bipoint,* c'est un couple de points. C'est tout. Par exemple, si A et B sont deux points de l'espace, (A, B) est un bipoint. On caractérise un bipoint par quatre choses :

- Son *origine,* c'est-à-dire son premier point A (attention de ne pas les inverser).

- Sa longueur, ou *norme :* c'est la distance entre A et B (mesurée en unités arbitraires).
- Sa *direction :* c'est une droite, la droite AB, passant par A et B.
- Son *sens :* de A vers B (pour le distinguer de (B, A)).

On dit que deux bipoints sont *équivalents* s'ils ont même norme, même sens et même direction : seule l'origine change.

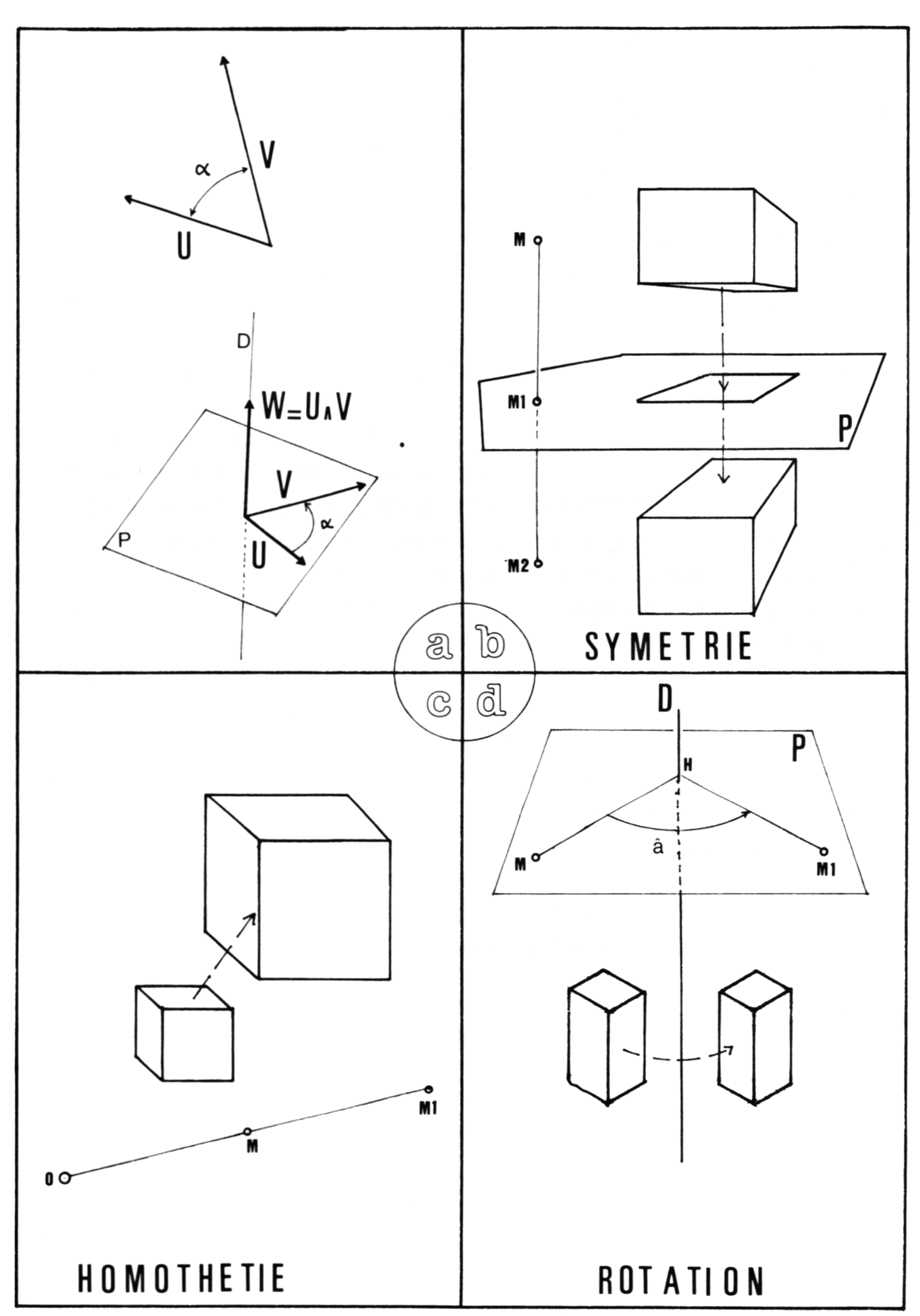
On appelle *vecteur* l'ensemble de tous les bipoints équivalents à un bipoint nommé (A, B) : c'est le vecteur **AB** que l'on note en caractères gras pour le distinguer de la droite AB.

On caractérise un vecteur par sa norme, sa direction et son sens qui sont, vous l'avez deviné, ceux du bipoint (A, B).

Cela vous paraît sans doute artificiel et bizarre, mais vous avez déjà utilisé des vecteurs. Ainsi, quand vous dites que le vent souffle à 80 km/h dans le sens nord-sud, et du nord, vous donnez ainsi la norme (80 km/h), la direction (axe nord-sud), et le sens (du nord vers le sud) d'un vecteur qui est ce que l'on appelle le vecteur-vitesse du vent.

Un vecteur peut être caractérisé par des coordonnées. En effet, soit **V** un vecteur donné, et O l'origine d'un repère O*xyz*. Il existe un point K tel que **OK** = **V** (vecteur **OK** égale vecteur **V**, ce qui signifie aussi que le bipoint (O,K) *appartient* à **V** : on dit que (O, K) est un *représentant* du vecteur **V**). Ce point K est unique. Il possède dans le repère O*xyz* trois coordonnées (x, y, z). *On les appelle aussi coordonnées de* **V**. Donc, pour résumer, les coordonnées d'un vecteur *sont celles du second point du représentant de ce vecteur placé à l'origine des axes.*

On ne peut pas vraiment dessiner un vecteur. On peut dessiner un de ses représentants. En mettant une flèche au bout, on montre qu'il s'agit bien de représenter un vecteur, comme sur la Figure II.D, schéma *a*.



V
α
U
D
W=U∧V
V
α
P
U
M
M1
M2
P
SYMETRIE
a
b
c
d
D
P
H
M
â
M1
M1
M
O
HOMOTHETIE
ROTATION

Figure II.D

Ce schéma est destiné à illustrer deux opérations sur les vecteurs, qui nous seront extrêmement utiles, surtout la première.

Soit deux vecteurs **U** et **V**, de coordonnées (u_0, u_1, u_2) et (v_0, v_1, v_2). On appelle produit scalaire de ces vecteurs le *scalaire* (c'est-à-dire le nombre, par opposition à un point ou un vecteur) :

$$\mathbf{U} \cdot \mathbf{V} = u_0 \cdot v_0 + u_1 \cdot v_1 + u_2 \cdot v_2$$

Ce produit scalaire est très intéressant car il donne le cosinus de l'angle α qui sépare les deux vecteurs. En effet, si u et v sont respectivement les normes de **U** et **V**, on a :

$$\mathbf{U} \cdot \mathbf{V} = u \cdot v \cdot \cos(\alpha)$$

Comme

$$u = \text{sqr}\,(u_0^2 + u_1^2 + u_2^2)$$

on obtient la formule :

$$\cos(\alpha) = \frac{(u_0 \cdot v_0 + u_1 \cdot v_1 + u_2 \cdot v_2)}{\text{sqr}\,((u_0^2 + u_1^2 + u_2^2) \cdot (v_0^2 + v_1^2 + v_2^2))}$$

qui permet de trouver le cosinus de l'angle séparant deux droites, comme nous le verrons.

Seconde opération, le produit vectoriel dont le résultat est un vecteur **W**. On le note $\mathbf{W} = \mathbf{U} \wedge \mathbf{V}$. Les coordonnées de **W** sont :

$$w_0 = u_1 \cdot v_2 - u_2 \cdot v_1 \,; \quad w_1 = u_2 \cdot v_0 - u_0 \cdot v_2 \,; \quad w_2 = u_0 \cdot v_1 - u_1 \cdot v_0$$

C'est un vecteur qui est perpendiculaire au plan qui contient **U** et **V**, et dont la norme est $w = u \cdot v \cdot \text{abs}\,(\sin(\alpha))$, ce qui permet de trouver le sinus de α (voir figure).

Notons encore une chose : lorsque **U** et **V** sont perpendiculaires ($\alpha = 90°$) leur produit scalaire est nul (car $\cos(90°) = 0$). S'ils sont parallèles, ($\alpha = 0$), c'est leur produit vectoriel qui est nul (car $\sin(0) = 0$) : en effet un vecteur de norme nulle est dit nul ; ses trois coordonnées sont égales à 0. C'est le cas du vecteur **AA**, pour n'importe quel point A.

5. APPLICATION DES VECTEURS : EQUATION D'UN PLAN, D'UNE DROITE

Soient trois points A, B, C. Il existe un unique plan P qui passe par ces trois points (à condition qu'ils ne soient pas alignés). Soit P ce plan, et M un point de coordonnées (x, y, z). Comment savoir si M appartient au plan ?

Il faut pour cela trouver l'équation du plan. Voici le raisonnement. Soit **U** et **V** les vecteurs **AB** et **AC**. P est le plan passant par A et *dirigé* par **U** et **V**. Soit $\mathbf{W} = \mathbf{U} \wedge \mathbf{V}$. Ce vecteur est perpendiculaire au plan P (voir figure). Il est donc perpendiculaire à toute droite contenue dans P, ou à tout vecteur. Or si M est contenu dans P, **AM** est un vecteur contenu dans P. Donc $\mathbf{AM} \cdot \mathbf{W} = 0$. C'est *l'équation* de P. Mettons-la en clair.

Le vecteur **AB** a pour coordonnées la différence des coordonnées de B et A, soit $((b_0 - a_0), (b_1 - a_1), (b_2 - a_2))$. En calculant aussi les coordonnées de **V**, on trouve celles de **W** :

$$w_0 = (b_1 - a_1) \cdot (c_2 - a_2) - (b_2 - a_2) \cdot (c_1 - a_1) \text{ ; etc.}$$

AM a pour coordonnées $((x - a_0), (y - a_1), (z - a_2))$; donc $\mathbf{AM} \cdot \mathbf{W} = 0$ s'écrit :

$$w_0 \cdot (x - a_0) + w_1 \cdot (y - a_1) + w_2 \cdot (z - a_2) = 0$$

ou encore :

$$\boxed{w_0 \cdot x + w_1 \cdot y + w_2 \cdot z - k = 0}$$

où k vaut :

$$w_0 \cdot a_0 + w_1 \cdot a_1 + w_2 \cdot a_2$$

Cette égalité est *l'équation* du plan P.

Pour une droite, le principe est à peine différent. Soient A et B deux points (distincts), et D la droite passant par A et B. Comment savoir si M est sur cette droite ? Très simple : si **U** est le vecteur AB, M est sur D si **AM**

est colinéaire (c'est-à-dire parallèle) à **U**. Cela se traduit comme ceci : il existe un nombre k tel que :

$$x = a_0 + k(b_0 - a_0)\,; \quad y = a_1 + k(b_1 - a_1)\,; \quad z = a_2 + k(b_2 - a_2)$$

Ces trois égalités sont appelées *équations paramétriques de la droite,* car elles dépendent d'un nombre k, qui est inconnu. Elles équivalent en général aux deux équations suivantes :

$$\frac{x - a_0}{b_0 - a_0} = \frac{x - a_1}{b_1 - a_1} = \frac{x - a_2}{b_2 - a_2} \quad (= k, \text{ d'ailleurs}).$$

Ces deux équations sont dites *intrinsèques,* car il ne s'y trouve aucun paramètre indéterminé.

Pour finir avec tout cela, quelques mots de vocabulaire. Un vecteur **W** perpendiculaire à un plan P est dit *normal* à ce plan. Un vecteur **U** joignant deux points distincts d'une droite D est dit *directeur* de la droite.

Enfin, si ces vecteurs sont *normés,* c'est-à-dire de norme un (ne pas confondre avec normal !), ils s'appellent le vecteur normal à P et le vecteur directeur de D, quoiqu'il y en ait deux : le vecteur en question, et son opposé (− **W** ou − **U**, c'est-à-dire le même vecteur, mais avec le sens inverse).

6. LES TRANSFORMATIONS DE L'ESPACE : PROJECTIONS, SIMILITUDES...

On appelle *application de l'espace* dans lui-même une fonction qui, à un point de coordonnées données (dans un repère fixe déterminé à l'avance), fait correspondre un autre point. Si (x, y, z) sont les coordonnées du point de départ M, et (x_1, y_1, z_1) les coordonnées du point d'arrivée M_1, il suffit d'écrire x_1, y_1 et z_1 comme des fonctions de x, y, z, pour avoir la transformation. Toutefois, il y a un autre moyen d'expliquer ce que fait telle ou telle transformation ou application de l'espace : l'expliquer géométriquement.

C'est ce que je compte faire dans ce paragraphe où nous allons étudier quelques transformations très simples de l'espace. Les calculs de coordonnées seront donnés au paragraphe suivant.

Tout d'abord, la plus simple est la *translation* de vecteur **V**. Le point (dit *image*) M_1 est tel que $MM_1 = \mathbf{V}$, tout simplement. En gros, cela consiste à déplacer les points dans une direction, et d'une distance données. Par exemple, un ballon poussé par le vent subit une translation.

Ensuite, la *projection*. Voyez la Figure II.D, schéma *b*. Un plan P étant donné, on appelle projection de M sur P le point M_1 tel que : M_1 est sur P et MM_1 est perpendiculaire à P. On voit aussi sur le dessin la projection d'une sorte de cube sur P (il en sera de même de toutes les transformations, pour les rendre plus claires). Notez que tous les points de la droite MM_1 ont M_1 pour projection sur P.

La *symétrie* est expliquée sur le même dessin. M_2 est le symétrique de M par rapport au plan P, ce qui veut dire que MM_2 est perpendiculaire à P (comme MM_1), et que les distances MM_1 et M_1M_2 (distance de M à P, et distance de M_2 à P) sont égales. Notez que M est aussi le symétrique de M_2. Si vous vous regardez dans un miroir, l'image que vous y voyez est votre symétrique par rapport au plan du miroir.

L'*homothétie* par rapport à un point O est telle que M_1 est sur la droite OM, et que $OM_1 = k \cdot OM$, où k est un nombre constant que l'on appelle rapport de l'homothétie ($k = 2$ sur la figure). Une homothétie grossit ou diminue les objets, mais garde leur forme et leur orientation inchangées. Cette transformation est illustrée par le schéma *c* de la Figure II.D.

Enfin, la *rotation* d'axe D et d'angle $\hat{a}$, visible sur le schéma *d* de la même figure, est telle que si P est le plan passant par M perpendiculaire à D, et H l'intersection de P et D, alors : M_1 est dans P ; la distance de M_1 à H est la même que de M à H ; et l'angle $\widehat{MHM_1}$ est égal à $\hat{a}$. Pour simplifier, une rotation consiste à faire tourner un objet.

Ces transformations peuvent se faire séparément, évidemment, ou bien à la suite. Si, par exemple, vous faites une rotation r puis une homothétie h, cela fait une certaine transformation que l'on appelle *composée de r par h* (et qu'on note $h \circ r$) (1).

On englobe sous le nom de similitudes toutes les applications qui sont les composées de symétries, de rotations, de translations et d'homothétie, car l'objet obtenu ressemble tout à fait à celui de départ. Par exemple, si vous prenez un stylo et que vous le jetez par terre, vous aurez fait sur lui une rotation-translation, donc une similitude.

Pour finir, sachez qu'il est possible d'inverser une application dans certains cas, c'est-à-dire d'en trouver l'inverse, l'application qui à M_1 fait correspondre M, au lieu du contraire.

(1) Lire : *h* ROND *r*.

Voici les inverses des transformations vues :

- L'inverse d'une translation de vecteur **V** est la translation de vecteur − **V**.
- Une projection n'a pas d'inverse.
- L'inverse d'une symétrie, c'est... elle-même (rappelez-vous, je vous ai dit que le symétrique de M_1 était M).
- L'inverse d'une homothétie de centre O et de rapport *k* est l'homothétie de centre O et de rapport 1/*k*.
- L'inverse d'une rotation d'axe D et d'angle $\hat{a}$ est la rotation d'axe D et d'angle $-\hat{a}$.

7. APPLICATIONS AFFINES ET MATRICES

On dit qu'une transformation de l'espace est affine si elle s'exprime sous la forme :

$$x_1 = m(1,1) \cdot x + m(1,2) \cdot y + m(1,3) \cdot z + c(1)$$

$$y_1 = m(2,1) \cdot x + m(2,2) \cdot y + m(2,3) \cdot z + c(2)$$

$$z_1 = m(2,1) \cdot x + m(3,2) \cdot z + m(3,3) \cdot z + c(3)$$

où les $m(i,j)$ (i et j variant de 1 à 3) et les $c(k)$ (idem pour k) sont des constantes qui caractérisent l'application affine. Nous allons voir que toutes les applications vues au paragraphe précédent sont affines.

Les trois relations ci-dessus s'expriment de manière plus simple sous la forme :

$$X_1 = M \cdot X + C$$

où M désigne la matrice contenant les coefficients $m(i,j)$, matrice à trois colonnes et trois lignes ; X_1, X et C sont les matrices contenant (x_1, y_1, z_1), (x, y, z) et les $c(k)$, respectivement : ce sont des matrices à une colonne et trois lignes.

Pour faire la somme de deux matrices, c'est tout simple, on additionne leurs coefficients terme à terme. Ainsi, la somme de M et de P (de coefficients $m(i,j)$ et $p(i,j)$ évidemment) est la matrice Q telle que $q(i,j) = m(i,j) + p(i,j)$, pour tout i et tout j. Cela exige que M et P aient autant de colonnes et de lignes, sinon la somme n'a pas de sens.

Le produit est un peu plus compliqué. Pour pouvoir le faire, il faut que le nombre n de colonnes de M soit égal au nombre de lignes de P. Alors la matrice $Q = M \cdot P$ a autant de lignes que M et autant de colonnes que P. Ses coefficients sont, si i est leur numéro de ligne et j leur numéro de colonne :

$$q(i,j) = m(i,1) \cdot p(1,j) + m(i,2) \cdot p(2,j) + ... + m(i,n) \cdot p(n,j)$$

Remarque capitale

$M \cdot P$ n'est pas forcément égal à $P \cdot M$, en admettant que les deux produits soient possibles, ce qui n'est pas sûr.

Vérifions que l'on peut calculer $M \cdot X + C$. Tout d'abord, M a trois colonnes, et X trois lignes. On peut donc calculer $M \cdot X$. Cette matrice a alors autant de lignes que M, soit trois, et autant de colonnes que X, soit une. C'est le cas également de C. On peut donc faire la somme : C.Q.F.D.

Une translation est telle que $M = Id$ (matrice Identité, telle que $Id \cdot P = P \cdot Id = P$, quelle que soit la matrice P)[1] et C est la matrice des coordonnées de V : $c(1) = v_0, c(2) = v_1, c(3) = v_2$. Donc $X_1 = X + C$, pour une translation.

Dès lors, toute application affine est composée d'une application linéaire (type $X_1 = M \cdot X$) et d'une translation (ajouter C au résultat).

Les autres applications vues sont linéaires. Nous allons en voir les matrices dans des cas particuliers.

Projection sur le plan xOy

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

(1) $Id = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Symétrie par rapport au plan xOy

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Homothétie de centre O et de rapport k

$$M = \begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & k \end{bmatrix}$$

Rotation d'angle $\hat{a}$ (c désigne le cosinus de $\hat{a}$, et s son sinus)

autour de Oz $$\mathrm{M} = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

autour de Oy $$\mathrm{M} = \begin{bmatrix} c & 0 & -s \\ 0 & 1 & 0 \\ s & 0 & c \end{bmatrix}$$

autour de Ox $$\mathrm{M} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix}$$

Si j'ai donné plus de détails sur la rotation, c'est qu'elle permet de trouver toutes les similitudes possibles. En effet, si l'on veut faire, par exemple, une symétrie autour d'un plan P autre que xOy, il suffit de trouver un repère dans lequel ce plan soit XOY (OXYZ étant le nouveau repère). On peut passer d'un repère à l'autre par une certaine rotation R (qui est elle-même le produit de deux ou trois rotations du type donné ci-dessus, en général). S étant la matrice de la symétrie par rapport à XOY, cette symétrie a pour matrice dans $Oxyz$: $\mathrm{M} = \mathrm{R}^{-1} \cdot \mathrm{S} \cdot \mathrm{R}$, où R^{-1} est

l'inverse de R, c'est-à-dire la matrice de la réciproque de la rotation (qui s'obtient en changeant les angles de signe). Nous aurons l'occasion de revenir sur ce point dès le prochain chapitre, par un exemple concret, ce qui sera sans doute plus clair.

Je pense qu'à présent vous êtes absolument paré pour aborder à fond le graphisme en trois dimensions.

III

L'ART DU VOYEUR OU COMMENT REGARDER SOUS TOUS LES ANGLES

Nous allons à présent passer aux choses sérieuses, c'est-à-dire commencer à raisonner et à programmer effectivement en trois dimensions.

Toutefois "qui veut voyager loin ménage sa monture" ; nous allons donc débuter par quelque chose de simple et facile à programmer, cela nous donnera d'autant plus de chances de parvenir à des choses nettement plus complexes.

Tout au long de ce livre, nous aurons besoin, d'une manière ou d'une autre, de *repérer* certains points dans l'espace. Nous utiliserons pour cela un repère (précisément), c'est-à-dire un point O fixé et trois axes Ox, Oy, Oz comme expliqué au Chapitre II. Par conséquent, pour commencer, nous allons tâcher de dessiner sur l'écran simplement ces trois axes, tout seuls.

Mais, direz-vous, cela ne sert à rien !

C'est tout à fait exact. Mais cela permettra de nous familiariser avec ce fameux repère, de mieux le percevoir dans l'espace, choses qui ne sont pas aussi simples qu'on pourrait le croire au premier abord. C'est pourquoi je vous déconseille vivement de sauter ce chapitre, ou de ne pas essayer les programmes qui s'y trouvent, même si ces notions vous paraissent simples, ou si elles vous sont familières : il n'y a là que du temps gagné.

1. TRACER LE REPERE SOUS UN POINT DE VUE DEFINI PAR DEUX ANGLES EN SPHERIQUES

Le titre de ce paragraphe vous a peut-être intrigué. Expliquons-le.

Ce n'est pas tout d'avoir l'intention de réaliser une vue d'un repère, ou plus généralement d'un objet. Il faut encore la réaliser en pratique. Et pour cela, votre ordinateur a d'abord besoin de savoir où se trouve l'observateur (c'est-à-dire vous-même) par rapport à l'objet qu'il est censé voir. On pourrait pour cela utiliser les points cardinaux (nord, est, etc.) mais ce n'est guère commode. Nous allons tout simplement utiliser les repères de l'espace qui sont représentés en Figure II.A, Chapitre II.

Tout d'abord, l'objet est supposé positionné en O, c'est-à-dire à l'origine du repère. Quant à l'observateur, jusqu'au Chapitre X, il sera supposé "infiniment loin", c'est-à-dire que le point Y où se trouvent ses yeux est à une distance de O beaucoup plus grande que la taille de l'objet.

Pourquoi cela ? Eh bien, il se trouve qu'ainsi, il suffit de projeter l'objet sur un plan E (écran) pour avoir une image qui paraisse *vraie,* alors que si l'objet était tout près, il serait déformé, comme sur ces photos où l'appareil a été placé tout près du nez du photographié : ces déformations engendrent les fameux points de fuites, de maniement très complexe. (Voir Chapitre X.)

Plutôt que la position exacte de Y, c'est donc le plan E, orthogonal (c'est-à-dire perpendiculaire) à la droite OY, qui nous intéresse. Or, ce plan est susceptible de bouger, puisqu'on veut voir notre objet sous des aspects divers. Il faut donc décrire sa position d'une manière ou d'une autre ; nous allons utiliser ici son vecteur normal.

Ce vecteur **N** est l'unique vecteur (au signe près toutefois) perpendiculaire au plan E et unitaire (c'est-à-dire, rappelons-le, dont la somme des carrés des coordonnées vaut 1). Appelons p, q, r ses coordonnées cartésiennes. Ces trois coordonnées sont aussi celles d'un point n tel que le vecteur **O*n*** soit égal à **N**. Et ce point n a aussi des coordonnées sphériques r, θ et ψ, que par abus de langage nous appellerons coordonnées sphériques du vecteur **N**.

Or, ce vecteur est unitaire, donc $r = 1$ (voir expression de r en fonction des coordonnées cartésiennes). Restent donc les deux angles θ et ψ, et maintenant le titre du paragraphe doit vous sembler plus clair.

On a donc les valeurs de p, q et r en fonction de ces angles :

$$p = \cos(\psi) \cdot \sin(\theta) \qquad q = \sin(\psi) \cdot \sin(\theta) \qquad r = \cos(\theta)$$

Vérifiez ainsi que $p^2 + q^2 + r^2 = 1$, sachant que $\cos^2(\hat{a}) + \sin^2(\hat{a}) = 1$ pour toute valeur de $\hat{a}$.

D'autre part, votre écran, c'est-à-dire le plan E, est doté par l'ordinateur d'un repère de coordonnées cartésiennes du plan (voir Figure II.A) que nous noterons O, X, Y pour éviter toute confusion.

Un point de coordonnées (x, y, z) étant donné, ainsi qu'une direction d'observation **N** de coordonnées (p, q, r), ce point a pour projection sur l'écran un point de coordonnées (X, Y).

Nous admettons les formules suivantes :

$$X = \frac{(q \cdot x - p \cdot y)}{R} \qquad Y = \frac{R \cdot z - r \cdot (p \cdot x + q \cdot y)}{R}$$

avec

$$R = \sqrt{p^2 + q^2}$$

formules qu'on ne saurait trouver simples.

D'où l'utilité des angles. En effet le lecteur vérifiera que $R = \sin(\theta)$. Dès lors, tout est simple :

$$X = x \cdot \sin(\psi) - y \cdot \cos(\psi)$$

$$Y = z \cdot \sin(\theta) - (x \cdot \cos(\psi) + y \cdot \sin(\psi)) \cdot \cos(\theta)$$

A présent, nous allons appliquer ces formules dans le Programme III.1 :

PROGRAMME III.1

```
400 '======================================== Trace des 3 axes du repere
410 c1=COS(Psi)
420 s1=SIN(Psi)
430 c2=COS(theta)
440 s2=SIN(theta)
450 CLS:ORIGIN 320,200
460 PLOT 0,0,3
470 TAG
480 DRAW 200*s1,-200*c1*c2:PRINT"x";
490 PLOT 0,0,3
500 DRAW -200*c1,-200*s1*c2:PRINT"y";
510 PLOT 0,0
520 DRAW 0,200*s2:PRINT"z";
530 TAGOFF
540 PEN 1
550 RETURN
```

Expliquons ce programme.

Il commence à la ligne 400. En effet, nous aurons encore à l'utiliser (n'oubliez pas de le conserver). Pour l'instant, en lignes 10 et suivantes, choisissez le mode et les couleurs (le dessin est fait en couleur 3), et mettez aussi un INPUT thêta et un INPUT psi.

En lignes 410 à 430, on appelle c_1, c_2, etc., les cosinus et sinus de nos deux angles, afin que l'ordinateur ne perde pas de temps à les recalculer plusieurs fois.

En 450, on efface l'écran et l'on place l'origine des coordonnées de l'écran O au centre de celui-ci. En 460, on y place aussi le plot, en choisissant à 3 la couleur de tracé.

En 470, le TAG permettra d'imprimer le nom des axes à leur extrémité.

Ensuite, on trace les axes proprement dits. Non pas en entier (ils sont infinis), mais en traçant un morceau compris entre O et un point situé sur l'axe à tracer, à une longueur donnée (ici 200). On applique donc les formules ci-dessus au point O (on obtient $X = Y = 0$), puis au point $(200, 0, 0)$ pour l'axe Ox, au point $(0, 200, 0)$ pour Oy, et au point $(0, 0, 200)$ pour Oz.

Le RETURN de la ligne 550 signale que le programme doit être appelé par un GOSUB. Cela vous permet de l'exécuter plusieurs fois en changeant les angles, avec un FOR-NEXT, ce que je vous conseille.

A présent, faites tourner votre programme, et regardez comme le repère évolue. N'hésitez pas à rajouter cette ligne :

```
545 LOCATE 1,25:PRINT "psi="psi;:PRINT "theta="theta;
```

et étudiez la position des axes en fonction des deux angles.

Vous serez aidé par la Figure II.A et par les exemples de la Figure III.A.

2. AUTRES DEFINITIONS DU POINT DE VUE

Nous avons choisi dans tout cet ouvrage le point de vue donné par deux angles en sphériques, pour des raisons de simplicité, quand c'était possible. Mais il nous arrivera d'avoir un point de vue donné simplement par un vecteur **N**′ de coordonnées p', q', r', même pas forcément unitaire.

Comment dès lors obtenir p, q et r ou, mieux, les deux angles θ et ψ ? On utilisera les formules suivantes : si

$$R = \text{sqr}\,(p'^2 + q'^2 + r'^2)$$

alors

$$p = \frac{p'}{R} \quad q = \frac{q'}{R} \quad r = \frac{r'}{R}$$

D'autre part :

$$\psi = \text{ATN}\left(\frac{q'}{p'}\right) \quad \text{et} \quad \theta = \text{ATN}\left(\frac{\text{sqr}\,(p'^2 + q'^2)}{r'}\right)$$

Ces formules nous seront bien utiles dans la suite ; elles sont données ici à titre indicatif.

Dans un autre ordre d'idées, si vous avez fait marcher le programme III.1, vous aurez remarqué que l'axe Oz reste toujours vertical, quelles que soient les valeurs des angles choisies : le repère est bien vu sous tous ses angles, mais il reste vertical, on ne peut le renverser.

Cela n'est possible qu'à condition de rajouter un troisième angle (mais oui !), noté φ (phi), et nommé rotation propre car il décrit la rotation de l'objet autour de l'axe visuel O-Observateur qui reste fixe, alors que les deux autres angles traduisaient un changement de cet axe visuel.

Les nouvelles coordonnées *projetées* de l'écran, notées X′ et Y′ se déduisent des anciennes X et Y par les formules de rotation du plan :

$$X' = X \cdot c_3 + Y \cdot s_3 \qquad Y' = -X \cdot s_3 + Y \cdot c_3$$

avec, vous l'aviez deviné, $c_3 = \cos(\varphi)$ et $s_3 = \sin(\varphi)$.

A vous à présent, si vous le souhaitez, d'adapter le Programme III.1, et tous ceux qui suivront pour pouvoir faire *tourner l'écran.*

Pour ce qui est de moi, mes axes Oz resteront verticaux !

3. APPLIQUER AUX AXES UNE TRANSFORMATION DE L'ESPACE

Nous avons vu au Chapitre II, Paragraphe 4, qu'il existait de nombreuses transformations de l'espace nommées similitudes (car le résultat après transformation ressemble à ce qu'on avait au départ), définies par des matrices.

Comment appliquer ces transformations à nos axes ou, plus tard, à d'autres objets ?

C'est tout simple, il suffit d'appliquer les matrices aux triplets (x, y, z), comme défini au Chapitre II.4.

En fait, nous l'avons déjà fait : pour trouver X et Y, nous avons appliqué à (x, y, z) la matrice M :

$$M = \begin{bmatrix} s_1 & -c_1 & 0 \\ -c_1 \cdot c_2 & -c_2 \cdot s_1 & s_2 \\ s_2 \cdot c_1 & s_2 \cdot s_1 & c_2 \end{bmatrix}$$

Nous avons obtenu un nouveau triplet (X, Y, Z) et nous n'avons pas tenu compte de Z.

A présent, vérifiez que M est le produit de ces trois matrices :

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & c_2 & s_2 \\ 0 & -s_2 & c_2 \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} c_1 & s_1 & 0 \\ -s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Reportez-vous au Chapitre II.4 et essayez de savoir exactement de quelles matrices il s'agit ; pouvez-vous expliquer cela à l'aide de la Figure II.A ? *(Question III-a.)* Sinon, n'utilisez plus de matrices par la suite, ou relisez le Chapitre II.4. De toute façon, nous en utiliserons le moins possible dans la suite.

Pour le moment, nous allons nous contenter de nos deux angles de sphériques ; ils suffiront largement à notre plaisir pendant un bon moment !

Plutôt que de devoir écrire de nouvelles valeurs de ces deux angles, ou que de les laisser varier sans contrôle, il est évidemment plus intéressant de pouvoir les changer soi-même, avec le clavier ou une manette de jeu.

C'est l'objet du Programme III.2, que nous aurons également l'occasion de réutiliser par la suite. Pour l'instant, nous allons l'additionner au Programme III.1 et remplacer les GOTO 'dessin' par des GOSUB 400.

PROGRAMME III.2

N.B.:L'indication 'dessin' signifie:"ligne où se trouve le dessin à exécuter" (par exemple la ligne 400 avec le programme III.1).

```
1300 '============================================== Choix clavier/manette
1310 CLS:PRINT"Choisissez entre:":PRINT
1320 PRINT"- Utiliser une manette de jeu (appuyez sur M)"
1330 PRINT"- Utiliser le clavier (appuyez sur C)"
1340 WHILE INKEY(38)=-1 AND INKEY(62)=-1:WEND
1350 IF INKEY(38)<>-1 THEN GOSUB 2700 ELSE GOSUB 2500
1360 CLS
1370 theta=45:psi=45
1380 GOTO 'dessin'
1900 '============================================= Test des touches reunies
1910 IF a=0 THEN b=10
1920 IF a=32 THEN b=20
1930 IF a=128 THEN b=5
1940 IF a=160 THEN b=45
1950 RETURN
2000 '================================================= Touches de rotation
2010 a$=INKEY$:IF a$="" GOTO 2010
2020 a=INKEY(k1):IF a<>-1 THEN GOSUB 1900:psi=psi-b:GOTO 'dessin'
2030 a=INKEY(k2):IF a<>-1 THEN GOSUB 1900:psi=psi+b:GOTO 'dessin'
2040 a=INKEY(k3):IF a<>-1 THEN GOSUB 1900:theta=theta-b:GOTO 'dessin'
2050 a=INKEY(k4):IF a<>-1 THEN GOSUB 1900:theta=theta+b:GOTO 'dessin'
2060 IF INKEY(79)<>-1 THEN CLS:END
2070 GOTO 2010
2500 '======================================== Indications pour le clavier
2510 CLS:k1=34:k2=27:k3=67:k4=59
2520 PRINT"Appuyez sur l'une de ces touches:"
2530 PRINT:PRINT"_ Q :seul,ou avec <CTRL>,ou avec"
2540 PRINT"<SHIFT>,ou avec les deux,pour avoir un"
2550 PRINT"basculement vers le bas de,respective-"
2560 PRINT"ment 10,5,20 ou 45 degres."
2570 PRINT:PRINT"_ W :seul,ou avec <CTRL>,ou avec"
2580 PRINT"<SHIFT>,ou avec les deux,pour avoir un"
2590 PRINT"basculement vers le haut de,respective-"
2600 PRINT"ment 10,5,20 ou 45 degres."
2610 PRINT:PRINT"_ P :seul,ou avec ...(idem) pour des"
2620 PRINT"rotations positives autour de l'axe z."
2630 PRINT:PRINT"_ O :seul,ou avec ...(idem) pour des"
2640 PRINT"rotations negatives autour de l'axe z."
2650 PRINT:PRINT"_ <DEL> pour arreter le programme."
2660 LOCATE 1,25:PEN 3:PRINT"Appuyez sur la barre pour commencer"
2670 WHILE INKEY(47)=-1:WEND
2680 RETURN
2700 '====================================== Indications pour la manette
```

```
2710 CLS:k1=74:k2=75:k3=73:k4=72
2720 PRINT"Appuyez sur l'une de ces touches:"
2730 PRINT:PRINT"_"+CHR$(241)+" :seul,ou avec <SHIFT> pour "
2740 PRINT"avoir un basculement vers le bas de,"
2750 PRINT"respectivement 10 ou 20 degres."
2760 PRINT:PRINT"_"+CHR$(240)+" :seul,ou avec <SHIFT> pour "
2770 PRINT"avoir un basculement vers le haut de,"
2780 PRINT"respectivement 10 ou 20 degres."
2790 PRINT:PRINT"_"+CHR$(242)+" :seul,ou avec <SHIFT> pour des"
2800 PRINT"rotations positives autour de l'axe z."
2810 PRINT:PRINT"_"+CHR$(243)+" :seul,ou avec <SHIFT> pour des"
2820 PRINT"rotations negatives autour de l'axe z."
2830 PRINT:PRINT"_ <DEL> pour arreter le programme."
2840 LOCATE 1,25:PEN 3:PRINT"Appuyez sur la barre pour commencer"
2850 WHILE INKEY(47)=-1:WEND
2860 RETURN
```

Ce Programme III.2 est assez parlant. Les lignes 1300 et suivantes vous permettent de choisir entre l'utilisation du clavier, moins commode mais plus souple car elle vous permet de choisir entre quatre valeurs d'incrément des angles (voir les instructions, lignes 2500 et suivantes), et celle de la manette, qui ne vous laisse que deux choix, la touche CTRL étant sans effet sur la manette.

Bien sûr, si vous n'avez pas de manette, ne tapez pas les lignes 1310 à 1350, ni les lignes 2700 et suivantes.

Les lignes 2000 et suivantes testent le clavier pour savoir quelles touches sont pressées. Si l'on appuie sur Q, W, O ou P pour le clavier, ou si l'on déplace la manette en *mode manette,* elles ajoutent une valeur b égale à 5, 10, 20 ou 45 (suivant que la touche a été pressée seule ou non — voir instruction INKEY en page 8.19 de votre manuel), ou la retirent, soit à θ, soit à φ selon la touche pressée.

Si ces lignes vous paraissent obscures, essayez tout de suite le programme en suivant les instructions qui s'affichent ; vous ne tarderez pas à comprendre.

Notez enfin que la ligne 1370 initialise les deux angles à 45°, sauf si vous avez oublié de mettre, avec le choix du mode en début de programme, l'instruction DEG !

Le programme va donc vous afficher en premier ce que vous voyez en Figure III.A, cadre 1. Cette configuration est dite isométrique car les trois axes ont alors la même longueur (compte tenu de ce que la lettre z a effacé une partie du troisième) ; les objets vus en isométrique sont plus harmonieux et plus faciles à se représenter dans l'espace, d'où le choix fait au Chapitre IX pour les surfaces en trois dimensions.

Appuyez à présent sur votre manette ou sur le clavier (suivant les instructions) ; faites varier vos angles.

Essayez d'arriver exactement aux mêmes configurations que celles de la Figure III.A (vous n'y parviendrez en fait qu'avec le clavier).

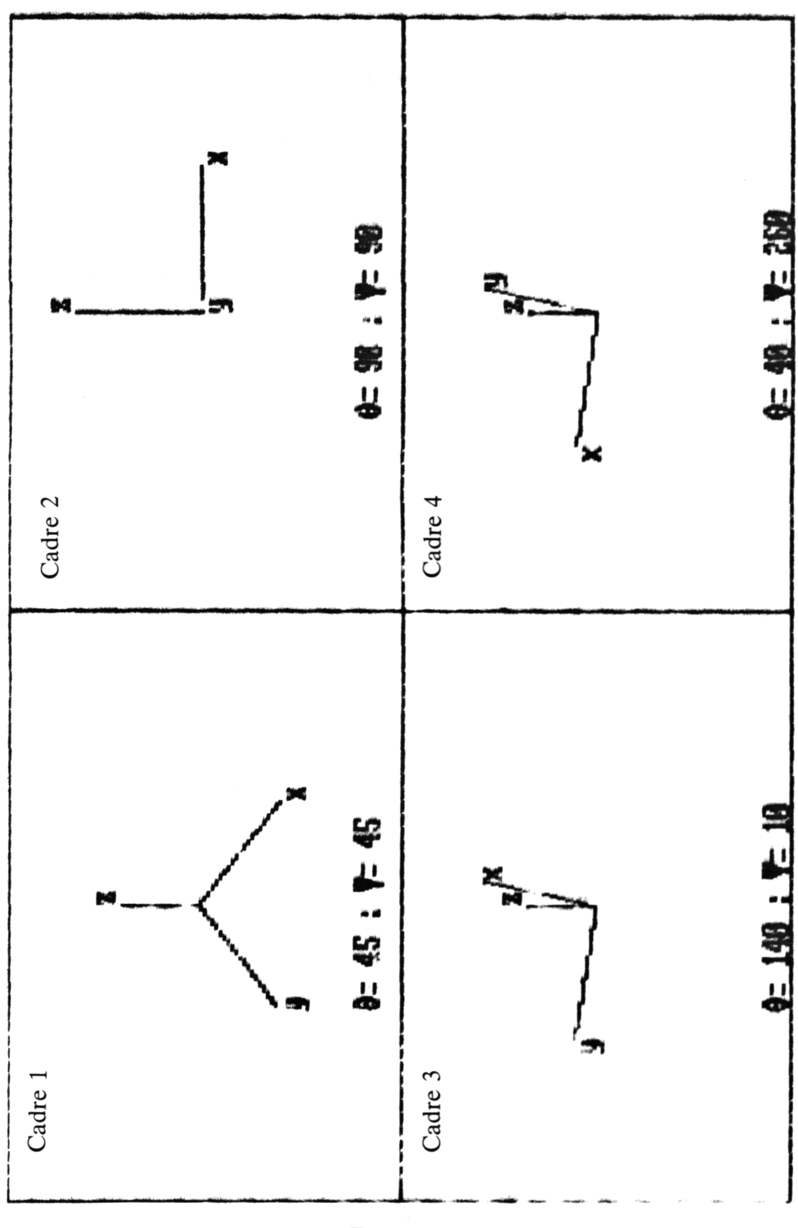


Figure III.A

D'ailleurs, jetons un coup d'œil sur cette figure. Dans le cas du cadre 2, avec quel plan E est-il confondu ? Quel est alors le vecteur normal ? Est-ce compatible avec les valeurs données des angles ? *(Question III.b.)*

Pour les cadres 3 et 4 de la même figure, les configurations des axes sont exactement les mêmes, mais x et y sont inversés. Or il se trouve que, d'une part, l'interversion de deux axes change un repère direct en un repère indirect[1] et, d'autre part, qu'une rotation ne change pas la direction d'un repère. Et nous n'avons fait que des rotations. Comment expliquez-vous ce mystère ? *(Question III.c.)*

Si vous avez réussi à répondre correctement à ces questions, bravo, vous êtes fin prêt à continuer. Sinon, amenez sur votre écran vos axes dans les mêmes configurations, progressivement. Et ne passez pas aux chapitres suivants sans avoir parfaitement compris les réponses !

(1) Un repère est indirect s'il n'est pas superposable à un repère direct.

REPONSES AUX QUESTIONS POSEES DANS CE CHAPITRE

Question III.a

La transformation consiste à changer le repère Oxyz en un repère OXYZ où OZ admet le vecteur **N** pour direction. Il faut pour cela exécuter une rotation d'angle ψ autour de Oz (matrice de droite). Après une symétrie qui intervertit les deux premiers nouveaux axes, on obtient un repère $OX_0Y_0Z_0$ (action de la matrice centrale). Enfin on exécute une rotation d'angle θ autour de OX_0 et l'on obtient le nouveau repère. L'ensemble est illustré par la Figure II.A, Om étant confondu avec OY_0 et OM avec OZ.

N.B. : N'oubliez pas que les transformations par les matrices sont effectuées en commençant par celle de droite.

Question III.b

Dans le cadre 2, E est confondu avec le plan xOz, normal à Oy ; donc $(p, q, r) = (0, 1, 0)$ et $\mathbf{N} = j$. Cela est compatible avec les valeurs données car $c_1 = c_2 = 0$ et $s_1 = s_2 = 1$.

Question III.c

Pour les deux autres cadres, les deux repères sont bien directs. Mais dans le cadre 3, l'axe Oz pointe vers le fond, alors que dans l'autre il pointe vers vous : des illusions d'optique et des inconvénients de la projection, n'est-ce pas...

IV

UN PLOT BALADEUR OU COMMENT TRACER N'IMPORTE QUOI DANS LE PLAN

Avant de poursuivre notre prospection tridimensionnelle, nous allons, le temps d'un bref chapitre, revenir au simple plan, pour étudier un petit "truc" qui nous sera bien utile dans la suite.

Il s'agit de ce que nous appellerons dans la suite le *plot,* du nom de l'instruction BASIC "PLOT" qui permet d'allumer un point lumineux sur l'écran.

1. TRACE SUR UN PLAN PAR UN DEPLACEMENT DE PLOT

En moyenne résolution (mode 1), votre Amstrad contrôle sur l'écran exactement 640 sur 100 points. Ces points sont appelés pixels. L'instruction PLOT *a, b, c* permet d'allumer le pixel de coordonnées *a, b* dans la couleur *c*. En particulier, si *c* est la couleur du fond *(paper)*, le pixel est en fait éteint : il disparaît. D'autre part, l'instruction test *a, b* permet de connaître la couleur du pixel de coordonnées *a, b* ; si cette couleur est celle du fond (0 en général), le pixel est éteint.

Ces instructions vont nous être extrêmement utiles. En effet elles permettent, avec un peu d'habileté et un bon programme, de tracer à peu près n'importe quoi dans le plan, et la Figure IV.A en est un bon exemple.

Ce dessin, comme tous ceux dont nous aurons besoin, est formé d'une succession de petits morceaux de droites. Son tracé correspond à l'allumage d'un certain nombre de pixels sur l'écran ; pour l'obtenir, il suffit donc de donner à l'ordinateur l'ensemble des points qui doivent être allumés.

Il va toutefois sans dire que, si l'on peut pour cela utiliser les coordonnées de ces points lorsque la figure est assez simple, ou lorsqu'elle est régie par une équation mathématique connue, il faudrait des heures et des heures de tâtonnements pénibles pour engendrer ainsi la tour Eiffel de la Figure IV.A, qui contient quelque 350 segments de droites.

Nous allons donc utiliser un moyen beaucoup plus simple. Pour enregistrer un point, il suffit en effet d'amener, sur l'écran, un point mobile (le fameux *plot baladeur*) à l'endroit exact où l'on souhaite qu'un pixel soit allumé.

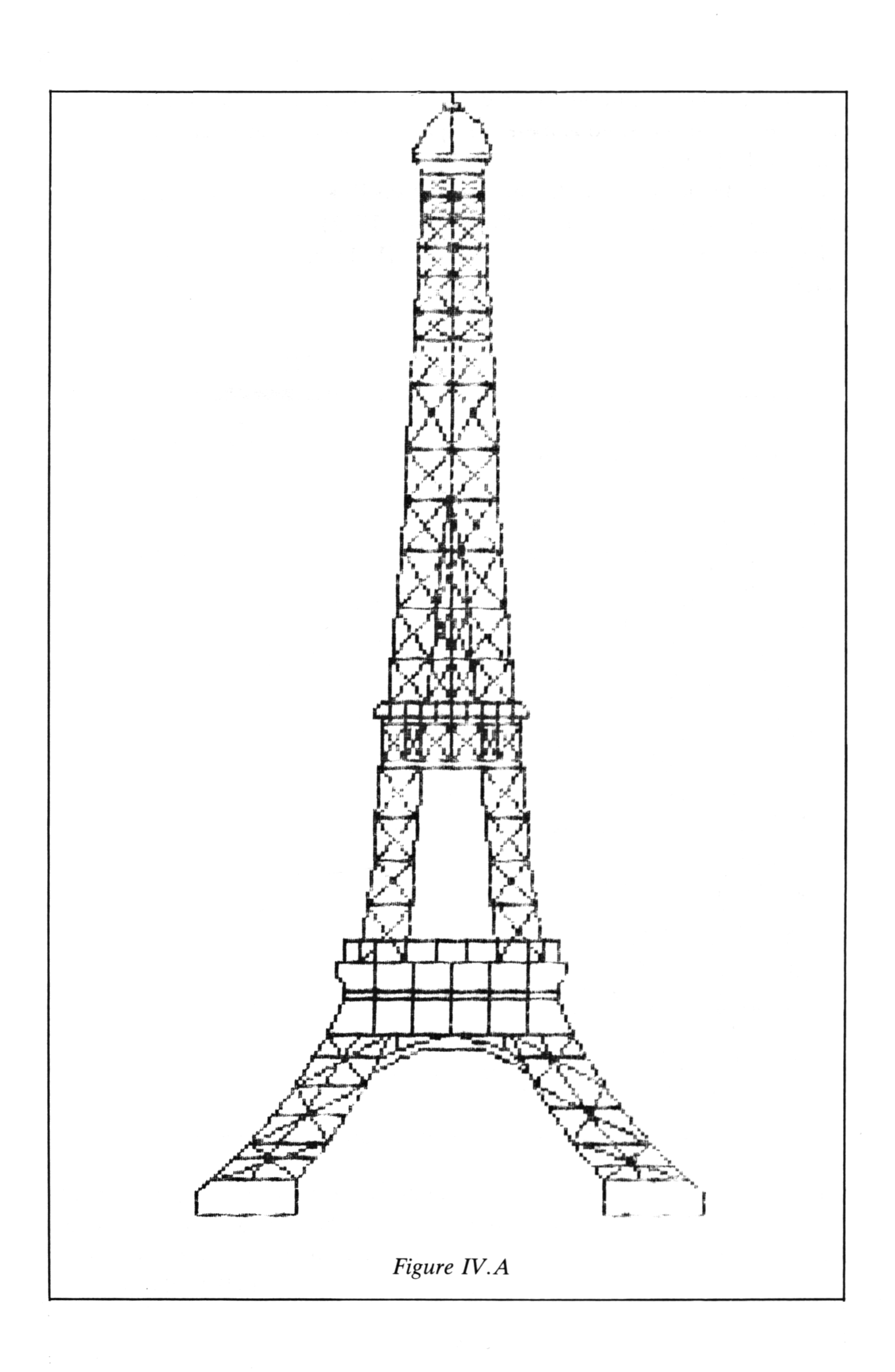


Figure IV.A

Pour déplacer commodément ce plot mobile, nous allons utiliser une manette de jeu ou, si vous n'en avez pas, le clavier (par exemple avec les touches fléchées).

Cela est facile, avec l'aide de l'instruction INKEY que nous avons déjà utilisée au chapitre précédent (Programme III.2).

Reportons-nous donc au Programme IV.1 : nous reconnaissons nos INKEY aux lignes 80 et suivantes.

PROGRAMME IV.1

```
10 '========================================================= Plot baladeur
20 MODE 1:INK 0,1:BORDER 1:INK 1,24:PEN 1:INK 2,6
30 DIM a(200,1)
40 DEFINT h,i,u,v,t
50 ORIGIN -1,-1
60 KEY 141,"goto 500"+CHR$(13):KEY DEF 24,1,141
70 PLOT 320,200,1:a(0,1)=1:t=1
80 WHILE  INKEY(79)=-1 AND INKEY(58)=-1 AND INKEY(74)=-1 AND INKEY(75)=-1
AND INKEY(72)=-1 AND INKEY(73)=-1 AND INKEY(68)=-1 AND INKEY(77)=-1:WEND
90 IF INKEY(68)<>-1 THEN a(t,0)=-1:GOTO 500
100 i=INKEY(74):IF i<>-1 THEN v=0:u=-2-i/16:GOSUB 300
110 i=INKEY(75):IF i<>-1 THEN v=0:u= 2+i/16:GOSUB 300
120 i=INKEY(73):IF i<>-1 THEN u=0:v=-2-i/16:GOSUB 300
130 i=INKEY(72):IF i<>-1 THEN u=0:v= 2+i/16:GOSUB 300
140 IF INKEY(79)<>-1 THEN PLOT XPOS,YPOS,0: t= t-1:a(t,1)=ABS(a(t,1)):
DRAW a(t,0),a(t,1):PLOT a(t,0),a(t,1),1:GOTO 180
150 IF INKEY(77)<>-1 THEN a(t,0)=XPOS:a(t,1)=YPOS:PLOT a(t-1,0),a(t-1,1),2:
DRAW a(t,0),a(t,1):t=t+1:PRINT CHR$(7):GOTO 180
160 IF INKEY(58)<>-1 THEN PLOT XPOS,YPOS,2:a(t,0)=XPOS:a(t,1)=YPOS:
a(t-1,1)=-a(t-1,1):t=t+1:PRINT CHR$(7):GOTO 180
170 IF t>199 THEN PRINT"Stop":END
180 LOCATE 1,1:PRINT USING "###";t
190 LOCATE 1,25:PRINT XPOS;:PRINT"/"YPOS;
200 GOTO 80
300 '================================================ Placement du Point
310 PLOT XPOS,YPOS,h
320 h=TEST(XPOS+u,YPOS+v)
330 PLOT XPOS+u,YPOS+v,1
340 RETURN
500 '========================================================= Restitution
510 CLS:MOVE 320,200
520 signe=SGN(a(0,1))
530 FOR t=1 TO 200
540 IF a(t,0)<0 GOTO 580
550 IF signe<0 THEN PLOT a(t,0),ABS(a(t,1)),1 ELSE
DRAW a(t,0),ABS(a(t,1)),1
560 signe=SGN(a(t,1))
570 NEXT
580 PRINT CHR$(7)
590 WHILE INKEY(47)=-1 AND INKEY(77)=-1:WEND
600 IF INKEY(77)<>-1 THEN t=t-1:GOTO 80
610 END
```

Il y a en plus une petite subtilité : en posant $i = \text{INKEY}(x)$, on obtient $i = 0$ ou $i = 32$ selon que l'on a appuyé sur la touche de n° x sans, ou avec, la touche SHIFT ; les paramètres u et v sont alors, selon les cas, égaux à -4, -2, 0, 2 ou 4. En outre, l'usage de la boucle WHILE-WEND fait que

l'ordinateur ne réagit qu'à certaines touches, ici les quatre directions de la manette (n°s 72 à 75), la touche "*e*" (n° 58), la touche DEL (n° 79), la touche TAB (n° 68) et la touche "fire 1" de la manette (n° 77).

Avant d'examiner de plus près ce programme, une petite explication est nécessaire.

2. ENREGISTREMENT DU TRACE : POINT PAR POINT, SOMMET PAR SOMMET

Avec notre plot, nous voulons *enregistrer,* c'est-à-dire conserver en mémoire un certain nombre de points de l'écran.

Pour cela, il suffit de créer une matrice, simplement nommée "*a*" dans le Programme IV.1, à deux colonnes et avec un nombre N de lignes. A la ligne n, on enregistre le n^{ieme} point, en plaçant son abscisse (sa première coordonnée) x dans la première colonne (n° 0) et son ordonnée y dans la seconde colonne (n° 1) :

$$a(n,0) = x : a(n,1) = y$$

Mais deux points de vue s'affrontent alors :

- soit on enregistre *tous* les points par lesquels on fait passer le plot ;
- soit on enregistre seulement *certains* points que l'on choisit soi-même, et l'ordinateur, lorsqu'il restitue le tracé, trace un segment entre chaque point enregistré.

Dans le premier cas, l'opération ci-dessus décrite est faite automatiquement, vous n'avez donc pas à vous en préoccuper.

Au contraire, dans le second, vous devez, chaque fois que vous souhaitez enregistrer un point, le signaler à la machine en appuyant sur une touche déterminée (par exemple la touche "*e*", comme enregistrer, ou le *fire* de votre manette de jeu).

Cette seconde méthode semble donc mauvaise et, de fait, elle l'est dans certains cas, en particulier si vous souhaitez dessiner à l'écran une forme assez complexe et tourmentée (par exemple un arbre) qui contient très peu de lignes droites.

Dans tous les autres cas, et en particulier dans ceux qui nous préoccuperont par la suite, c'est au contraire la seconde méthode qui est la meilleure pour plusieurs raisons.

Tout d'abord, la mémoire de votre ordinateur a beau être vaste, le nombre N de lignes de la matrice *a* est nécessairement fini, et par conséquent le nombre de points enregistrés aussi (on peut aller jusque vers 3 000) ; il convient donc, surtout pour des dessins assez complexes comme la tour Eiffel, de se restreindre. Or, imaginez que vous ayez à enregistrer un simple carré de 40 pixels de côté. Le périmètre de ce carré étant de $4 \times 40 = 160$ points, la première méthode enregistrera 160 points, mais la seconde seulement 4, car il n'y a que quatre sommets dans un carré ! De plus, si vous dotez votre dessin d'un grand cadre qui suit le pourtour de l'écran, la seconde méthode n'emploiera toujours que 4 points, mais la première quelque $2 \times (640 + 200) = 1\,680$ points ! Vous voyez qu'il ne vous restera pas grand-chose de vos 3 000 points possibles pour le dessin en lui-même !

Cela seul suffirait à faire choisir la seconde méthode, mais il y a en plus une seconde bonne raison. Examinez les segments de la tour Eiffel de la Figure IV.A : vous voyez que leur pente, c'est-à-dire leur inclinaison par rapport à l'horizontale, est très variable. Cela prouve que ce dessin a été obtenu avec la seconde méthode, car la première n'autorise qu'un petit nombre de pentes. En effet, puisqu'elle enregistre tous les déplacements de plot, et que le plot ne peut se déplacer que verticalement ou horizontalement (à la rigueur en oblique si vous appuyez sur deux touches à la fois), chaque petit segment ne peut qu'être orienté de même ; un segment de pente quelconque devra donc être représenté de manière approximative par une succession de petites *marches* toutes égales, comme pour un escalier, ce qui n'est ni esthétique ni commode. Par contre, avec la seconde méthode, il suffit de placer le plot successivement aux deux extrémités du segment à tracer et d'enregistrer, en vous moquant totalement du chemin qu'il prend pour passer de l'un à l'autre.

Peut-être comprenez-vous aussi maintenant pourquoi j'ai affirmé que la première méthode était quand même valable pour les dessins tourmentés : s'il y a très peu de segments de droites sur le dessin, les deux défauts disparaissent *de facto*.

Dans la suite, nous utiliserons tout de même la seconde méthode, que vous maîtriserez facilement avec un peu d'habitude.

3. EXPLICATION DU PROGRAMME IV.1 : LEVER LE CRAYON

A présent, le Programme IV.1 doit vous sembler plus clair : après avoir fixé les couleurs en 20 (le plot baladeur, couleur 1, sera jaune pour qu'on le distingue bien, le tracé des segments se fera en rouge), on dimensionne l'arrangement *a* pour qu'il puisse contenir 200 points (cela nous suffira dans la suite). La touche contenant le signe £, à côté de CLR, est réglée pour que vous puissiez revoir votre dessin par sa seule pression (ligne 60). Puis le plot est placé au centre de l'écran, et le compteur *t* est initialisé à 1. Ensuite, l'ordinateur attend que vous pressiez une touche (ligne 80).

Si vous appuyez sur une touche de déplacement (par exemple, la n° 72, aller vers le haut), alors le programme donne une certaine valeur aux paramètres *u* et *v* (ici 0 à *u* et 4 ou 2 à *v*) et passe en 300. Là, il remet le pixel où le plot se trouve à sa couleur initiale *h* (faute de quoi votre écran ne tarderait pas à se trouver envahi de petits points jaunes, et vous ne sauriez plus lequel est votre plot baladeur), puis enregistre avec TEST la nouvelle valeur de *h*, c'est-à-dire la couleur du pixel où l'on va placer le plot (lignes 320 puis 330), toujours en jaune. Vous constatez qu'avec *u* nul et *v* positif, comme dans notre exemple, le plot est bien légèrement déplacé vers le haut.

Après l'exécution du sous-programme 200, le programme arrive à la ligne 170 (sauf si vous appuyez sur deux touches à la fois) où il vérifie que *t* ne dépasse pas la valeur maximale de dimensionnement de la matrice *a*, faute de quoi il arrête le programme. Ensuite, il affiche la valeur de *t* (ligne 180) et, en bas, les coordonnées du plot (cela servira dans d'autres chapitres).

Puis il retourne en 80, attendant que vous appuyiez sur une touche.

Si vous appuyez sur TAB (n° 68), le dessin est fini : il vous le restitue alors en 500. Si vous appuyez sur *fire* (n° 77), il enregistre le point où se trouve le plot, et trace un trait qui le relie au précédent (ligne 150).

D'autre part, vous avez droit à l'erreur. Si vous appuyez sur DEL (n° 79), il décrémente de 1 le compteur *t*, et replace le plot à l'emplacement de l'avant-dernier point enregistré, non sans tracer une ligne noire sur le dernier segment rouge tracé (ne vous inquiétez pas si l'effacement n'est que partiel, c'est sans importance) ; ainsi, les prochains points enregistrés remplaceront ceux que vous aurez rejetés, à cause de la décrémentation de *t*.

Soit, mais à quoi sert la touche "*e*" (n° 58) ? Eh bien, c'est tout simple. Nous avons vu que, lorsqu'on enregistre un point avec la touche *fire*, un segment était tracé entre les deux derniers points enregistrés. Mais lorsque vous dessinez, il vous arrive de *lever* votre crayon : entre le nouveau point, où vous reposez votre crayon, et l'ancien, où la mine à quitté le papier, il n'y a pas de segment de tracé. Il faut que la machine vous permette de faire de même ; c'est le but de l'utilisation de cette touche *e*. Comme la touche *fire*, elle stocke dans le fichier les coordonnées du plot comme un nouveau point à retenir, mais elle ne trace aucun segment ; le point reste isolé tant que vous n'aurez pas réutilisé la touche *fire*.

Mais un problème va se poser lors de la restitution du dessin ; en effet, comment le programme de restitution (lignes 500 et suivantes) saura-t-il s'il faut tracer un segment ou non ?

Là encore, il faut enregistrer. L'idée qui vient naturellement à l'esprit est de rajouter à la matrice a une colonne remplie de *flags* : -1 s'il faut tracer un segment jusqu'au point suivant, 1 sinon, par exemple.

Seulement voilà, cela augmente de 50 pour 100 la place prise en mémoire par cette matrice !

Pour éviter ce qui pourrait devenir un désagrément, nous allons nous en tirer par une astuce, et laisser la ligne 30 en l'état.

L'astuce se trouve en ligne 50, qui vous a peut-être intrigué. Le fait de placer l'origine des axes hors de l'écran fait que tous les points de l'écran ont des coordonnées de signe constant, ici strictement positives. Dès lors, le *signe* de ces coordonnées va nous servir de *flag*. Si le signe de la seconde coordonnée est négatif, et non positif comme il devrait l'être puisque les coordonnées du plot sont toujours positives, c'est que l'on a enregistré ce point avec la touche "*e*" (voyez en ligne 160 le changement de signe de $a(t-1, 1)$, donc il ne faut pas tracer de segment. Cela vous explique le test de la ligne 550 qui porte sur le signe de la seconde coordonnée du point *précédent* (c'est $a(t-1, 1)$ dont on change le signe). C'est aussi pour cette raison que $a(0, 1)$ a été initialisé à une valeur strictement positive, sinon il serait nul, donc en 550 on tracerait un segment entre le premier point et l'origine.

Résumons l'astuce :

signe de $a(t, 1)$ positif : Tracer un segment jusqu'au point suivant.

signe de $a(t, 1)$ négatif : Aller au point suivant sans tracer de segment.

Au fait, diront les puristes, pourquoi changer le signe de $a(t-1, 1)$ et non celui de $a(t, 1)$, ce qui permettrait d'utiliser le point n° 0 ? C'est bien entendu tout à fait possible, mais en fait cela ne simplifie nullement le programme, surtout au niveau de la restitution, car cela exige de faire revenir le plot en arrière. Le gain est donc, à mon avis, nul. Cela étant, il ne s'agit que d'un détail.

A propos de détails, notez encore ceux-ci sur ce programme :

- La même astuce est réutilisée sur la première coordonnée pour signifier au programme de restitution que le dessin est fini (voir lignes 90 et 540).
- Lorsque la restitution est finie, le programme se met en attente (ligne 590) ; si vous appuyez sur la barre (n° 47), tout s'achève en 610 ; si vous appuyez sur *fire*, le programme retourne en 80, vous permettant de reprendre votre dessin où vous l'aviez laissé.
- J'ai trouvé plus commode qu'un bip sonore se fasse entendre à l'enregistrement de chaque point, d'où le PRINT CHR$(7) des lignes 150 et 160.
- Lorsqu'un point est enregistré seul, il disparaît momentanément. Pour ne pas le perdre de vue, rajoutez l'instruction $h = 1$ en ligne 160.

A présent, faites tourner le programme. Vous allez constater sur votre Amstrad quelque chose de curieux: les coordonnées de vos points augmentent de 2 en 2, même si vous avez tapé $u = -1 - i/36$, etc., en lignes 100 et suivantes. C'est qu'en moyenne résolution, le plot ne fait pas un pixel mais quatre. C'est pourquoi vous ne pourrez obtenir des coordonnées impaires. C'est aussi pourquoi le plot avance par multiples de 2 ; le nombre de points parcourus ne s'en trouve pas réduit, mais le temps de déplacement est divisé par deux.

Si toutefois vous trouvez le plot un peu lent dans ses déplacements, n'hésitez pas à utiliser la touche SHIFT, pour aller deux fois plus vite ; nous avons défini i pour cela.

4. USAGE DES COULEURS

Nous aurons besoin, dans les chapitres suivants, non seulement de ce

plot baladeur (alors n'oubliez surtout pas de conserver le Programme IV.I), mais en plus de pouvoir l'utiliser en couleurs.

En effet, jusqu'à présent, la couleur du dessin était uniforme. Maintenant cela ne sera plus le cas, si vous rajoutez au Programme IV.1 le supplément IV.1 *bis*. Vous pourrez ainsi changer la couleur du tracé (ou sa nuance si vous avez un moniteur monochrome : dans ce cas, choisissez les couleurs de telle sorte qu'on les distingue bien les unes des autres).

PROGRAMME IV.1 bis

```
20 MODE 1:INK 0,1:BORDER 1:INK 1,24:PEN 1:INK 2,6:INK 3,18
30 DIM a(200,2)
40 DEFINT h,i,u,v,t,c
70 PLOT 320,200,1:a(0,1)=1:t=1:c=2
80 WHILE  INKEY(79)=-1 AND INKEY(58)=-1 AND INKEY(74)=-1 AND INKEY(75)=-1
AND INKEY(72)=-1 AND INKEY(73)=-1 AND INKEY(68)=-1 AND INKEY(77)=-1 AND
INKEY(62)=-1:WEND
95 IF INKEY (62)<>-1 GOTO 400
150 IF INKEY(77)<>-1 THEN a(t,0)=XPOS:a(t,1)=YPOS:PLOT a(t-1,0),a(t-1,1),c:
DRAW a(t,0),a(t,1):a(t,2)=c:t=t+1:PRINT CHR$(7):GOTO 180
160 IF INKEY(58)<>-1 THEN PLOT XPOS,YPOS,c:a(t,0)=XPOS:a(t,1)=YPOS:a(t,2)=c
:a(t-1,1)=-a(t-1,1):t=t+1:PRINT CHR$(7):GOTO 180
400 '======================================== Changement de couleur
410 LOCATE 1,1:PRINT"      "
420 LOCATE 1,1:INPUT "Couleur";c
430 c=c MOD(4)
440 PLOT XPOS,YPOS,c
450 LOCATE 1,1:PRINT"          "
460 GOTO 180
550 IF signe<0 THEN PLOT a(t,0),ABS(a(t,1)),a(t,2) ELSE
DRAW a(t,0),ABS(a(t,1)),a(t,2)
```

Ce supplément IV.1 *bis* est assez clair, je pense :

- En ligne 20, on choisit la troisième couleur (vert ici) en plus.
- En ligne 30, on rajoute une colonne à la matrice, colonne qui contiendra le numéro de la couleur. Cette couleur, *c*, est initialisée au rouge, comme sur le Programme IV.1, en 70.
- La touche C vous permettra d'aller au sous-programme de changement de couleur en 400 (lignes 80 et 95). Ce sous-programme affiche à l'emplacement du compteur *t* la question "Couleur ?". Vous devez alors lui donner le numéro de la couleur que vous souhaitez utiliser. Cette valeur (comprise entre 0 et 3, d'où la ligne 430) restera fixe jusqu'à ce que vous réappuyiez sur la touche C.
- Enfin, n'oublions pas que le tracé doit se faire dans la couleur que vous avez demandée, aussi bien lors du dessin que lors de la restitution, d'où le changement des lignes 150, 160, et 550.

J'ai déjà dit que nous nous servirions de ce programme, je vous propose donc de le conserver (mais séparément). Cela ne vous empêche pas, bien sûr, d'en profiter d'ici au chapitre suivant ; vous verrez que ses possibilités sont très étendues, et que l'on peut ainsi faire de très jolies choses !

V

L'ART DE L'ILLUSION OU COMMENT REPRESENTER UN DESSIN SUR UN PLAN INCLINE

Nous pénétrons à présent pour de bon dans l'univers des effets graphiques en trois dimensions, car il faut bien reconnaître que les trois axes du Chapitre III étaient assez innocents.

Pourtant, c'est encore de plan que nous allons parler. Mais ce plan, nous allons à présent le considérer non comme un univers en soi, mais comme une partie de l'univers réel, de l'espace tridimensionnel.

1. L'ILLUSION DE LA PERSPECTIVE

Nous avons déjà vu au Chapitre III que la perspective dite *cavalière*, qui consiste en une simple projection, en fait n'était qu'une approximation des perspectives *vraies*, d'usage beaucoup plus complexe, que nous verrons au Chapitre X.

Mais les figures qui illustrent ce chapitre sont là pour vous prouver que cette perspective a beau être approximative, elle n'en est pas moins très réaliste.

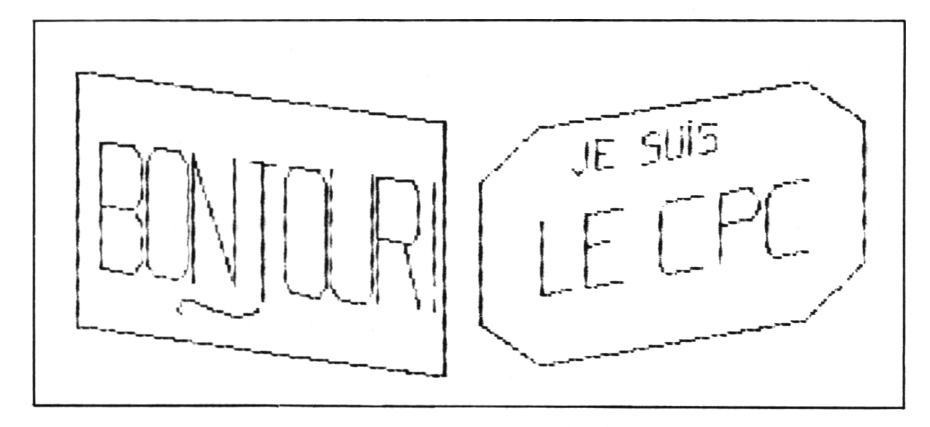


Figure V.A

Car la perspective est l'essence même de cet art de l'illusion qu'est le dessin sous toutes ses formes ; devant elle, on a du mal à considérer le dessin tel qu'il est, c'est-à-dire plan, et l'on cède à l'illusion de la profondeur.

Cela ne facilite pas toujours le travail du dessinateur, car toute inexactitude dans la perspective risque d'être vite perçue. C'est pourquoi il serait risqué, et de plus difficile, d'obtenir des images semblables à celles de ce chapitre, directement à l'aide du plot baladeur du Chapitre IV. Peu importe, la machine est là pour faire le travail à notre place. Là encore, il suffit d'avoir la bonne idée.

Réfléchissons... Que voulons-nous faire ?

Avant de passer à la représentation de volumes, nous voulons essayer de représenter dans l'espace un objet plan, non pas situé sur un plan parallèle à l'écran mais sur n'importe quel plan, même incliné.

Il s'agit donc, en fait, de passer d'un plan P (celui à dessiner) à un autre plan E (l'écran), via l'espace tridimensionnel.

Eh bien, nous savons déjà le faire.

En effet, soit un point (a, b) de P. Suivant la position de P dans un repère R qui est fixe par rapport à lui (alors que le repère R_0 que nous avons utilisé jusqu'à présent est fixe par rapport à l'écran E), il correspond au point (a, b), un point de l'espace de coordonnées (X, Y, Z) dans le repère R, et (x, y, z) dans R_0.

Or, on peut choisir R comme on le veut, pourvu qu'il soit fixe par rapport à P. Nous le choisissons simplement, par exemple de telle sorte que P soit égal au plan XAZ (A étant l'origine de R). Dès lors, $X = a$, $Y = 0$, $Z = b$. Donc, au point (a, b) on fait correspondre dans l'espace le point $(a, 0, b)$; c'est le contraire d'une projection.

Maintenant on veut trouver (x, y, z). Cela est tout simple, si l'on considère que les deux repères se déduisent l'un de l'autre par une rotation d'angle ou d'axe donné. Ici, nous allons considérer que l'on passe de l'un à l'autre par la rotation dont la matrice M a déjà été donnée au Chapitre III.3, Paragraphe 3. On applique donc cette matrice au triplet $(a, 0, b)$, et l'on obtient le triplet (x, y, z). Enfin, l'on se souvient que l'écran E est parallèle au plan xOy du repère R_0 ; la projection de (x, y, z) sur E a pour coordonnées (x, y) ; il est donc inutile de calculer z.

Nous sommes ainsi passés du doublet (a, b) de P au doublet (x, y) de E ; c'était ce que nous voulions.

Tout cela est bien plus facile à mettre en œuvre qu'à expliquer. Le Programme V.1 l'exécute en deux lignes (1030 et 1040). Le lecteur attentif y reconnaîtra les coefficients de la matrice M (mais seulement 4 sur 9, car $Y = 0$ et l'on ne calcule pas z).

Avant de taper ce programme, il faut aller chercher les précédents. Tapez :

LOAD" Programme III.2"	Sur ce programme changez "dessin" en 1000 (5 corrections à faire).
RENUM 2000	L'ordinateur vous signale qu'il ne connaît pas la ligne 1000, marquée 5 fois. C'est normal.
MERGE "Programme IV.1"	Pour rajouter ce programme au précédent.

A présent, tapez les onze lignes du Programme V.1.

PROGRAMME V.1

```
610 ORIGIN 320,200:DEG:t0=t-1
620 GOTO 2000
1000 '============================================= Dessin du Plan incline
1010 CLS:signe=SGN(a(0,1))
1020 FOR t=1 TO t0
1030 x=SIN(psi)*(a(t,0)-320)/1.8
1040 y=(-COS(psi)*COS(theta)*(a(t,0)-320)+SIN(theta)*(ABS(a(t,1))-200))/1.8
1050 IF signe<0 THEN PLOT x,y,1 ELSE DRAW x,y,1
1060 signe=SGN(a(t,1))
1070 NEXT
1080 PRINT CHR$(7)
1090 GOTO 2150
```

Il ne vous reste qu'à faire tourner le programme. Le début se présente comme celui du plot baladeur ; de fait, tout sera exactement semblable jusqu'au moment où, après la restitution, le dessin fini, vous appuierez sur la barre d'espacement. La machine enregistre alors la valeur maximale $t0$ du compteur t (cela évitera bien des tests), se place en degrés et recentre l'origine avant de vous envoyer en 2000 (lignes 610, 620), c'est-à-dire sur le Programme III.2 que nous avons déjà utilisé.

Je ne détaille pas la suite : en dehors du calcul de x et y dont nous avons déjà parlé, les lignes 1000 et suivantes sont très proches des lignes 500 et suivantes du Programme IV.1 pour une raison évidente mais fondamentale : un segment est toujours transformé en un autre segment par des rotations et des projections, et au total cela se traduit tout de même par un DRAW.

Le mieux est, de toute façon, comme toujours, que vous essayiez afin de vous rendre compte par vous-même. Je ne vous donnerai qu'un petit conseil : mettez un cadre rectangulaire autour de vos dessins, cela augmente l'illusion.

Notez également que rien ne vous empêche de taper MERGE "Programme IV.1 *bis*", et de corriger la ligne 1050 en remplaçant les 1 par des $a(t,2)$, pour obtenir des dessins en couleurs.

Enfin, la division par 1,8 des valeurs naturelles de x et y est faite pour éviter que votre dessin déborde de l'écran quand il est en oblique. Mais si votre dessin est petit, supprimez-la, sinon vous ne verriez plus grand-chose.

2. RECTO VERSO

Le jeu de cartes lancé en l'air de la Figure V.B prouve que notre Programme V.1 (uni aux autres) peut déjà être utile dans des jeux. Nous en verrons aussi des applications plus "sérieuses" lors des chapitres suivants.

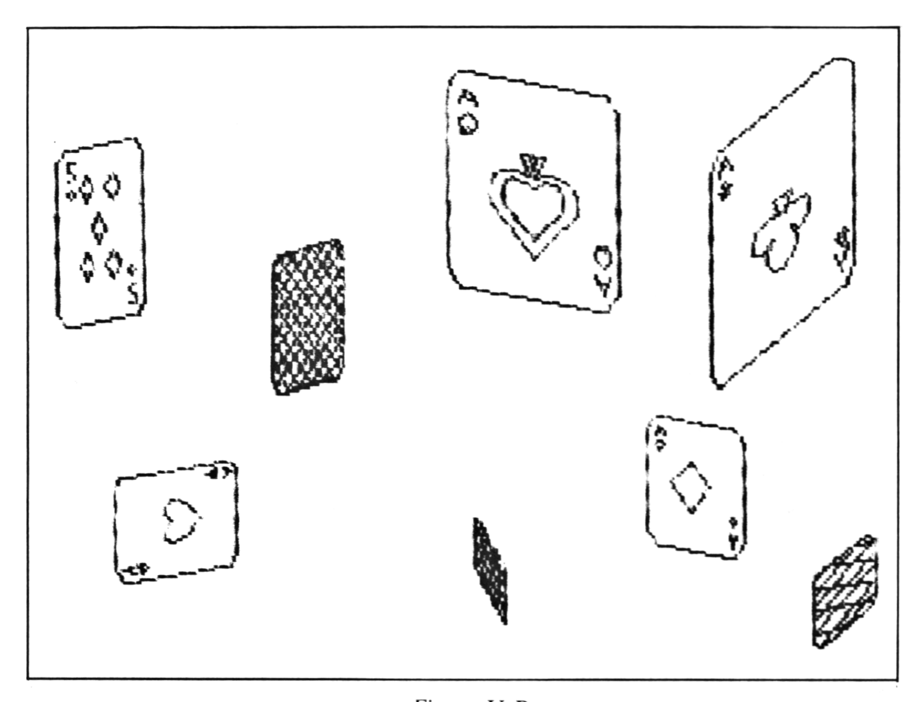


Figure V.B

Cependant, si l'on veut réaliser une animation avec une carte, on ne saurait s'en tenir au seul Programme V.1

En effet une carte a *deux* faces. Mais si vous avez essayé le Programme V.1, vous avez probablement constaté que, lorsqu'on "retourne" le plan P en le faisant suffisamment tourner, la seconde face est semblable à la première, vue dans un miroir. En quelque sorte, le programme considère que vous avez dessiné sur une sorte de plaque transparente que l'on verrait alors de dos, et non sur une feuille de papier sur les deux faces de laquelle on peut faire des dessins différents.

Le Programme V.1 *bis*, greffé sur le précédent, vous permettra d'engendrer cet effet de recto verso.

PROGRAMME V.1 bis

```
5 DIM a(400,1):u0=0:GOSUB 10:t0=t-1:u0=200:GOSUB 10:t1=t-1:ORIGIN 320,200:
DEG:GOTO 2000
30 '
70 PLOT 320,200,1:a(u0,1)=1:t=1+u0
170 IF t-u0>199 THEN PRINT"Stop":END
520 signe=SGN(a(u0,1))
530 FOR t=1+u0 TO 200+u0
610 RETURN
620 '
1005 s=SGN(SIN(Psi)):IF s<0 THEN u0=200:v0=t1 ELSE u0=0:v0=t0
1010 CLS:signe=SGN(a(u0,1))
1020 FOR t=u0+1 TO v0
1045 x=s*x:y=s*y
```

Ce programme est peu compliqué. En 5, on envoie deux fois de suite sur le plot baladeur, transformé en sous-programme appelé par GOSUB : la première fois pour le recto, la seconde pour le verso. Les deux dessins sont enregistrés à la suite l'un de l'autre dans la matrice *a* dont la capacité a été doublée à cet effet, le premier à partir de 0, le second à partir de 200. N'oubliez pas d'éliminer la ligne 30, sinon vous verrez s'afficher "Array already dimensionned in 30". On retrouve aussi en ligne 5 les ordres de la ligne 620 qui a été éliminée. Bien entendu, on procède à quelques réajustements dans la boucle (70 à 610), très clairs, je pense. Reste à comprendre comment l'ordinateur va savoir de quel côté du plan on se trouve, afin de déterminer s'il doit afficher le recto ou le verso.

Pour cela, vous remarquerez (essayez) que lorsque psi est multiple de pi, c'est-à-dire lorsque sin (psi) = 0, l'on voit le plan P par la tranche ; on est alors à la limite entre le recto et le verso. De fait, si vous êtes familiarisé avec les sinus, sachant que $P = xOz$, vous vérifierez sur la Figure II.A que, lorsque le sinus de psi change de signe, on passe du recto au verso, ou l'inverse.

Dès lors *s*, le signe de sin (psi), sert d'indicateur. On a décidé arbitrairement ici que lorsqu'il était négatif on était au verso, et sinon au recto, d'où les lignes 1005 et 1020.

Encore une remarque sur la ligne 1045. Elle correspond à une symétrie centrale du verso. Si vous ne la mettez pas, vous verrez votre verso comme dans un miroir, comme pour le Programme V.1 quand on est du mauvais

côté du plan (et il n'y a pas de raison que cela change). En fait, ici votre dessin sera redevenu normal mais il sera tête-bêche par rapport au recto. En effet, le seul vrai moyen pour avoir un recto et un verso ordonnés comme sur une feuille de papier sans se forcer à faire un dessin à l'envers, est de faire, pour t supérieur à 200, $a(t, 0) = 320 - a(t, 0)$, c'est-à-dire une symétrie dans P autour de la droite centrale. Cependant une telle opération, au demeurant tout à fait faisable (mais supprimez alors la ligne 1045), vous obligera à faire du calcul mental si vous souhaitez mettre un cadre autour de vos dessins comme pour les cartes, afin que les deux cadres soient bien ajustés.

C'est pourquoi je n'ai pas donné le programme sous cette forme, car cela le complique, et bien souvent pour rien. Ainsi, pour mes cartes à jouer, peu m'importe que le verso soit tête-bêche : il est symétrique.

Et maintenant, tout se complique...

VI

LES POLYEDRES SOUS TOUS LES ANGLES

Nous sommes déjà à plein dans l'espace, mais les objets que nous avons jusqu'à présent cherché à y représenter n'étaient pas encore à proprement parler tridimensionnels, en ce sens qu'ils pouvaient tous être contenus dans un plan.

Toujours avec la volonté d'augmenter progressivement la difficulté, nous allons à présent nous attaquer à de véritables objets tridimensionnels, des objets que, même en poussant fort, nul ne peut faire rentrer dans un plan, mais tout d'abord aux plus simples de ces objets, à mon sens, à savoir les polyèdres, et en particulier les polyèdres convexes.

1. QU'EST-CE QU'UN POLYEDRE ?

Question évidemment essentielle si l'on souhaite savoir de quoi l'on parle. Je ne me permettrais pas de douter que mon aimable lecteur sache de quoi il en retourne, mais afin d'éviter toute erreur, mieux vaut se montrer précis sur une chose aussi capitale.

Donc, définissons. Un polyèdre est tout d'abord un *solide*. Par là, entendons notamment qu'il est *compact*, c'est-à-dire surtout *borné* (aucune de ses dimensions n'est infinie) et accessoirement *fermé* (si on le considère comme creux, il n'y a pas de trou dans ses parois), et de plus *connexe*, c'est-à-dire en un seul morceau (une pomme et une poire font *deux* solides, pas un seul). Tout cela peut paraître évident mais ce n'est pas une raison pour le passer sous silence. Un polyèdre est donc un solide, mais de plus, *limité de toutes parts par des plans*. Sa surface extérieure est une succession de morceaux de plans, que l'on appelle *faces*. De ces définitions il résulte que ces faces sont elles-mêmes des *polygones*, c'est-à-dire des lignes brisées refermées sur elles-mêmes. Chaque pliure de ces lignes brisées porte le nom de *sommet*, et entre deux sommets, on a éventuellement une *arête*.

Voici donc définis les polyèdres. Reste à savoir à quoi ils peuvent bien servir. Notons au préalable que nous connaissons déjà de nombreux polyèdres : un dé (un cube) est un polyèdre, les pyramides en sont aussi, et les diamants de bijouterie sont aussi taillés en polyèdre. En outre, il existe de nombreux objets qui sont peu différents de polyèdres, et ainsi pourrons-nous étendre notre champ d'action à des maisonnettes, des meubles, des chars, ...

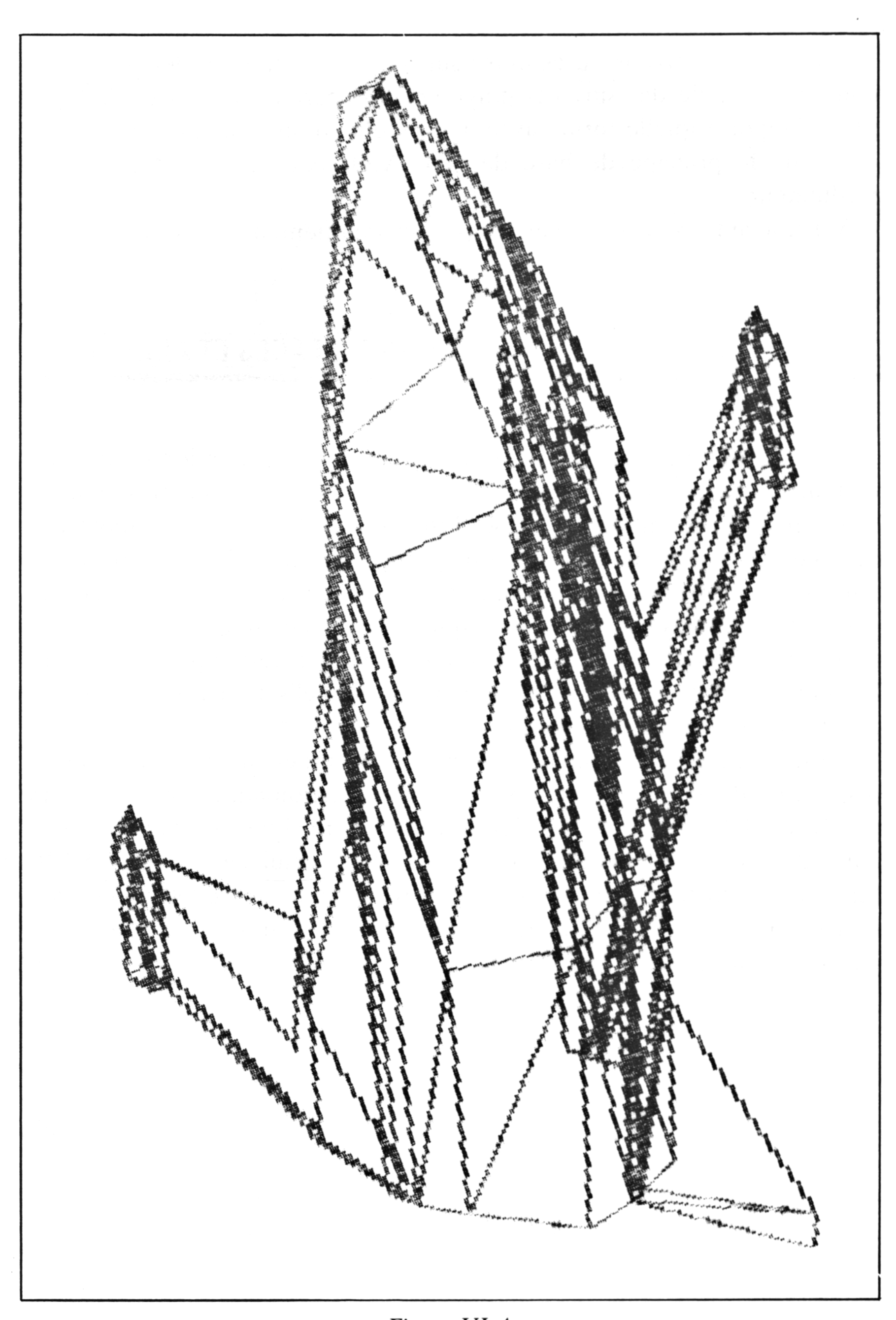


Figure VI.A

Enfin, nous verrons au chapitre suivant que la théorie des polyèdres, combinée à celle des surfaces polynomiales, permet de concevoir à peu près n'importe quelle forme avec une excellente approximation. C'est là d'ailleurs le principe de base de la C.A.O. (Conception Assistée par Ordinateur).

Mais, avant d'y rêver, revenons plus prosaïquement à nos polyèdres.

2. DEFINITION DU POLYEDRE : SOMMETS, FACES ET ARETES

Nous avons vu qu'un plan peut être défini par une équation ou par son vecteur normal, indifféremment. Pour le polyèdre, le même problème de la nature de la définition se repose. J'insiste tout d'abord sur l'importance capitale de ce problème. N'oublions pas que l'objet même de l'informatique est de traduire dans le langage spécial de la machine des choses parfois très simples pour nous, mais qui ne le sont en fait que parce qu'elles sont exprimées dans un langage courant, sans signification pour l'appareil. De fait, ce problème de définition ou de modélisation est, de loin, le problème majeur de la technique informatique.

Ainsi, vous avez certainement compris, éventuellement à l'aide de la page précédente, d'un exemple ou d'un dictionnaire, ce qu'est un polyèdre. D'ailleurs, nous n'utiliserons guère que le cube comme illustration de ce paragraphe, un cube comme celui de la Figure VI.B. Mais votre Amstrad, qui sommeille paisiblement sur votre table, n'en sait rien encore... Comment lui faire comprendre ? Sûrement pas avec les phrases alambiquées du dictionnaire...

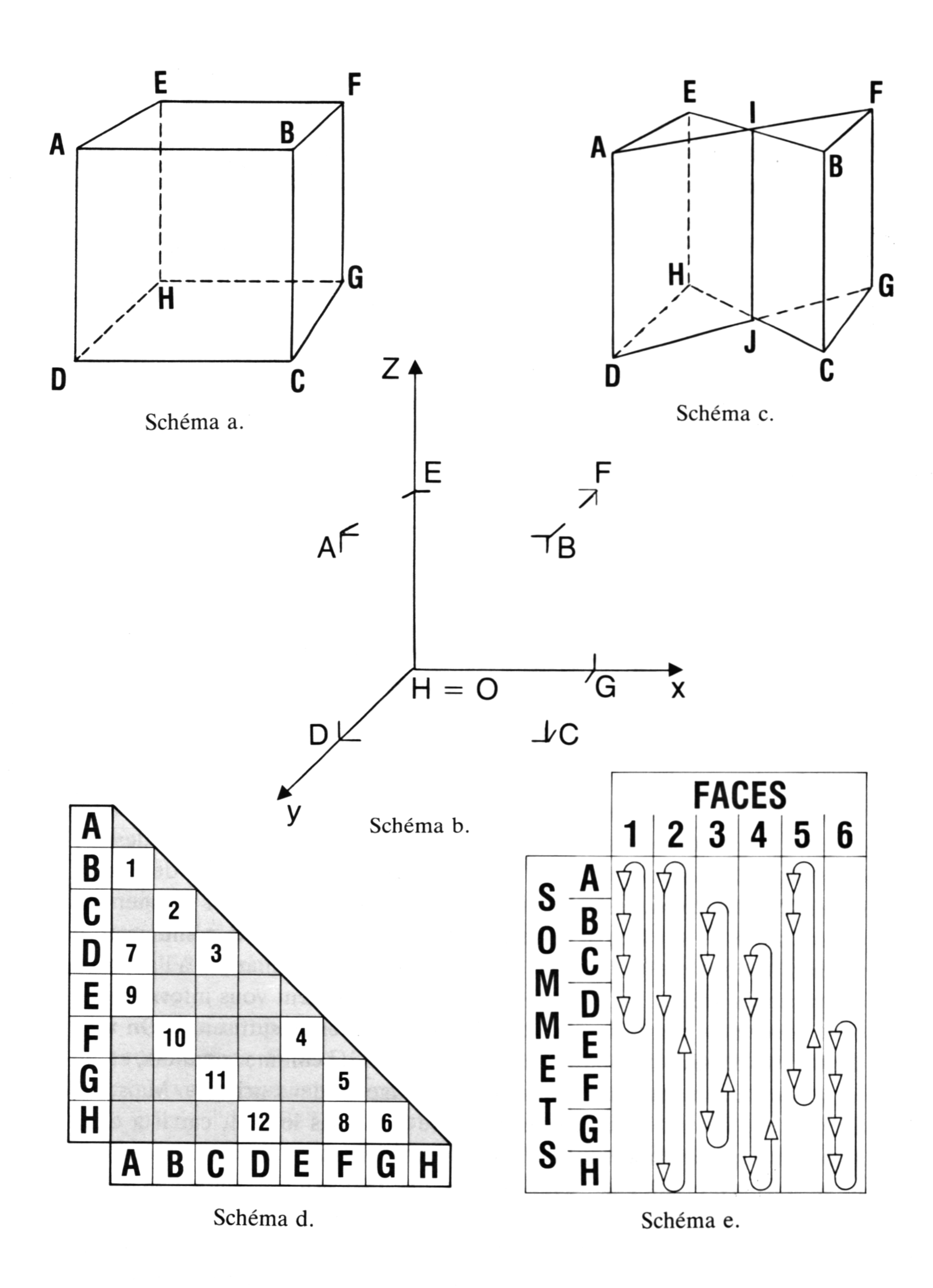


Schéma a.

Schéma b.

Schéma c.

Schéma d.

Schéma e.

Figure VI.B.a,b,c,d,e.

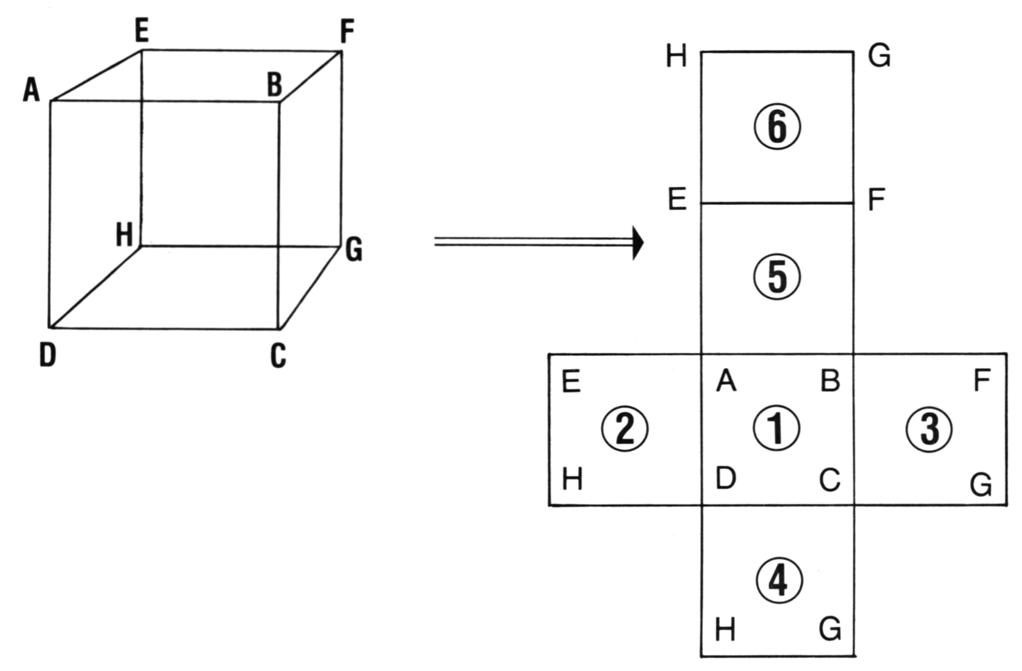


Schéma f.

Figure VI.B.f.

Reportons-nous à la Figure VI.B et à ses schémas successifs. Le schéma *a* donne l'objet à représenter : le cube ABCDEFGH, ici dessiné en perspective avec point de fuite (perspective réelle). Pour définir ce cube, l'idée qui vient naturellement à l'esprit est de lui donner les coordonnées de ses huit sommets en choisissant un repère, comme sur le schéma *b*, où A est (0, 1, 1), B (1, 1, 1), etc., puis de demander à l'appareil de tracer les arêtes. Le schéma *c* vient immédiatement vous informer que cette idée n'est pas la bonne, ou plutôt qu'elle est insuffisante. On a ici dessiné un polyèdre qui admet aussi ABCDEFGH comme sommets, et qui n'est pas un cube mais une sorte d'assemblage de deux prismes. Mais, me direz-vous, je vois bien que ce polyèdre n'est pas le bon, car il a deux sommets supplémentaires, I et J. Soit, vous le savez, mais l'ordinateur, lui, ne le sait pas, et allez donc lui expliquer ce qu'est un sommet...

Nous saurons d'ici peu la différence essentielle entre ces deux polyèdres : le premier est convexe, le second ne l'est pas. Or il n'y a qu'un seul polyèdre convexe qui passe par nos huit points. Cette propriété nous sera utile, mais pas pour le moment car je n'ai pas encore l'intention de me

limiter aux polyèdres convexes. J'aurais très bien pu décider de tracer le polyèdre du schéma *c* au lieu du cube, et dans ce cas, les points seuls ne sauraient suffire. D'ailleurs c'est très simple : seuls les tétraèdres (à quatre sommets) peuvent être définis par la donnée de leurs seuls sommets ; ce, pour la bonne raison que deux sommets quelconques d'un tétraèdre sont toujours reliés entre eux par une arête. Ce n'est déjà plus vrai pour un polyèdre à cinq sommets.

A la donnée des sommets, il faut donc nécessairement ajouter au moins un renseignement supplémentaire.

Le tableau *d* est un renseignement supplémentaire possible, on donne ici à l'ordinateur les arêtes à tracer. Elles sont numérotées (arbitrairement) de 1 à 12. On voit alors quelle serait la démarche à suivre : dans un premier temps, une matrice (mettons SOM) serait dimensionnée à $s-1$ lignes et 2 colonnes (s désigne le nombre de sommets, donc ici 8). SOM (u, v) serait la $v+1^{\mathrm{e}}$ coordonnée du $u+1^{\mathrm{e}}$ point (n'oubliez jamais que les matrices contiennent des lignes 0). Ainsi SOM $(0,0)$ serait la première coordonnée du premier point (le point A), c'est-à-dire 0, etc. Puis une seconde matrice (mettons AR), dimensionnée à $s-1$ lignes et autant de colonnes, comme le tableau *d*, serait remplie de *flags* : AR $(u, v)=0$ si les points $u-1$ et $v-1$ ne sont pas reliés par une arête, 1 dans le cas contraire. Ainsi, ici on aurait AR $(3,4)=0$ (D et E non reliés), mais AR $(4,5)=1$ (E et F reliés). Le reste s'en déduirait par des tests. Autre possibilité, plus simple : AR est dimensionné à $(a-1,1)$ ($a=$ nombre d'arêtes $=12$ ici). Chaque ligne contient une paire de points, extrémités d'une même arête. Par exemple, AR $(0,0)=0$ (numéro de A) et AR $(0,1)=1$ (numéro de B) représentent la première arête, AB. Cela permet d'économiser de la place mémoire, et du temps d'exécution (en évitant les tests), mais je trouve cette seconde possibilité un peu plus pénible à utiliser ; cela ne l'empêche pas d'être la meilleure.

Je n'utiliserai pourtant pas cette méthode dans ce chapitre, pour plusieurs raisons. Primo, elle est suffisamment simple à mettre en œuvre pour que vous puissiez vous passer de moi, même si vous êtes débutant ; la seule partie "difficile" est la projection, mais elle sera de toute façon exposée. Secundo, nous l'utiliserons un peu plus loin, sous une forme légèrement différente. Tertio, malgré sa simplicité, elle est encore trop compliquée dans certains cas. Quarto, elle est totalement inadaptée au problème épineux de la gestion des faces cachées.

C'est précisément en vue de ce problème que nous allons utiliser une méthode différente et plus complexe.

Il s'agit d'une méthode basée sur les *faces* d'un polyèdre. Pour mieux voir ces faces, le schéma *f* représente le cube "éclaté", c'est-à-dire coupé le long de certaines arêtes pour le rendre plan. Les six faces du cube sont alors numérotées (arbitrairement aussi). Le tableau *e* recense les points contenus dans chaque face, mais d'une manière un peu particulière. En effet, chaque face a ici quatre sommets. Or, si l'on enregistre la cinquième face, par exemple, dans l'ordre ABEF, et que l'on exécute le programme qui sera donné immédiatement, on obtiendra une aberration du type de celle du schéma *c*. Il faut donc enregistrer ABFE, pour avoir effectivement une face carrée. C'est ce que signifie le petit *circuit* qui figure sous le nombre 5 dans le tableau *e*. Ce circuit signifie également, outre qu'il faut prendre les sommets dans l'ordre des flèches, que le choix du premier sommet est arbitraire ; la face 5 est aussi bien FEAB ou BFEA que ABFE. En outre, je tiens à préciser tout de suite que si le sens de circulation n'est pas tout à fait arbitraire pour la gestion des faces cachées, le programme se chargera de rétablir le bon sens, donc vous pouvez aussi bien choisir AEFB que ABFE.

Il existe, comme avec les arêtes, plusieurs moyens de mémoriser les faces de notre cube. Le moyen qui sera utilisé ici n'est sans doute pas le meilleur, quoiqu'il soit fort simple ; mais j'ai déjà dit que les programmes de ce chapitre sont élaborés en fonction du problème des faces cachées car, encore une fois, c'est sans cela le système par arêtes qui me semble le plus simple.

Nous allons donc enregistrer notre polyèdre sous la forme d'une matrice, simplement dénommée *a* dans la suite. Cette matrice a trois dimensions. Soit *f* le nombre de faces du polyèdre, S le nombre *maximal* de sommets (ou d'arêtes) par face. Alors, *a* sera dimensionné en $(f-1, S-1, 2)$. Le nombre $a(t, u, v)$ désignera la v^{ieme} coordonnée (x pour $v=0$, etc.) du u^{ieme} sommet de la t^{ieme} face. Là encore, je devancerai les critiques : ce n'est pas le meilleur système, tant s'en faut. En particulier, chaque sommet devra être enregistré plusieurs fois puisqu'il appartient à plusieurs faces. Toutefois, cette méthode est en fait très commode pour certains polyèdres particuliers pour lesquels elle est d'une extraordinaire simplicité, et totalement inutilisable, j'en conviens, pour les autres ; je donnerai alors une autre méthode.

En attendant, nous supposerons, dans le paragraphe suivant, que la matrice *a* (matrice des faces) a été constituée, d'une manière ou d'une autre. Rien ne vous empêche d'ailleurs de la constituer "à la main" pour le

cube par exemple. Cela aura le mérite de vous décourager à tout jamais de le refaire, et vous accueillerez ainsi avec joie les méthodes des paragraphes suivants.

3. DESSIN ET DEPLACEMENT DU POLYEDRE (UTILISATION PAR LES FACES)

Avant d'utiliser le Programme VI.1, exécutez les instructions suivantes :

LOAD "ProgrammeIII.1".DELETE 400.RENUM 1050
MERGE "ProgrammeIII.2".DELETE1350-1380, 2070, 2680, 2860 -
Corrigez le mot "dessin" en 600 (dans les 5 lignes).
MERGE "ProgrammeVI.1".

PROGRAMME VI.1

```
470 '============================================== Calcul de l'echelle
475 INK 3,6
480 FOR t=0 TO f-1
490 FOR u=0 TO AA(t)-1
500 r=a(t,u,0)*a(t,u,0)+a(t,u,1)*a(t,u,1)+a(t,u,2)*a(t,u,2)
510 IF r > rmax THEN rmax=r
520 NEXT:NEXT
530 e=200/SQR(rmax)
540 theta=45:psi=45
600 '=============================================== Trace du Polyedre
610 GOSUB 1000
620 FOR t=0 TO f-1
635 FOR u=0 TO AA(t)-1
640 x=a(t,u,0)
650 y=a(t,u,1)
660 z=a(t,u,2)
670 GOSUB 1200
680 IF u=0 THEN a=XX:b=YY:PLOT a,b,1:GOTO 700
690 DRAW XX,YY,1
700 NEXT u
710 DRAW a,b,1
720 NEXT
730 GOTO 2000
1000 '=============================== Point de vue de Projection et axes
1200 '=================================== Calcul des coordonees Projetees
1210 XX=(s1*x-c1*y)*e
1220 YY=(s2*z-c1*c2*x-s1*c2*y)*e
1230 RETURN
1350 IF INKEY(38)<>-1 GOTO 2700 ELSE GOTO 2500
2680 GOTO 300
2860 GOTO 300
15000 '================================================= Verification
15010 FOR t=0 TO f-1:FOR u=0 TO AA(t)-1:FOR v=0 TO 2:PRINT a(t,u,v):NEXT
:NEXT:WHILE INKEY$="":WEND:PRINT:PRINT:NEXT
15020 END
```

Quelques brèves explications sur ce Programme VI.1. Les lignes 470 et suivantes sont destinées à calculer le facteur d'échelle e. Ce facteur, calculé une fois pour toutes et réutilisé aux lignes 1210 et 1220 à chaque tracé, est calculé de telle manière que, quel que soit le polyèdre rentré au départ, il tiendra tout entier sur l'écran, et plus précisément dans un cercle de rayon 200 autour du centre de celui-ci. Les angles, d'autre part, sont initialisés en isométrique.

Au sujet du tracé du polyèdre (lignes 600 et suivantes), notez tout d'abord la présence de la matrice AA que je n'ai pas encore mentionnée. Cette matrice, à une seule ligne de f éléments, contient tout simplement, pour chaque face, le nombre de sommets qu'elle comporte. Elle est destinée à raccourcir l'exécution du programme dans le cas d'un polyèdre où le nombre de sommets par face serait très variable. Dans le cas contraire, AA (t) peut être simplement remplacé par S, nombre maximal de sommets par face. D'autre part, l'utilisation des variables x, y, z et la séparation des lignes 1200 et suivantes du corps du programme sont de pure commodité (voir Chapitre X). Rien ne vous oblige à les conserver.

Le reste du programme principal est assez clair. Notons que les lignes 680 et 710 sont nécessaires pour éviter les liaisons intempestives au changement de face et pour refermer les faces sur elles-mêmes. Pour ce qui est du calcul de XX et YY, nous retrouvons exactement le même calcul qu'au chapitre précédent.

Quant aux lignes 15000 et suivantes, elles vous serviront à vérifier que votre matrice a est correcte. Permettez-moi d'avance de vous adresser mes plus sincères félicitations si vous réussissez à ne pas en avoir besoin !

Les autres lignes étant de simples liaisons internes du programme, il ne vous reste plus qu'à l'exécuter. Si vous ne voulez plus des axes, entrez la ligne :

```
1095 RETURN
```

et votre polyèdre sera seul sur l'écran. Pour ses déplacements, ils sont ceux du Chapitre III.

4. POLYEDRES CONVEXES ET CONCAVES. AVANTAGES, INCONVENIENTS

Quoique le Paragraphe 5 du présent chapitre ne s'y limite pas totalement, je me vois obligé, pour la bonne compréhension de ce chapitre, de parler dès à présent des polyèdres convexes.

Tout d'abord, il s'agit au préalable de les définir. On dit, d'une façon générale, qu'une partie de l'espace est *convexe* si, lorsqu'on y prend au hasard deux points *a* et *b*, le segment [*ab*] tout entier est contenu également dans la partie. Ce n'est là qu'une des nombreuses propriétés des parties convexes ; ces propriétés se retrouvent naturellement dans les polyèdres convexes, et même s'y multiplient. En particulier, notons que les faces des polyèdres convexes sont des polygones (comme pour tout polyèdre) également convexes (cette propriété est parfois également vérifiée par des polyèdres *concaves*, c'est-à-dire non convexes ; elle n'est donc pas suffisante). D'autre part, un polyèdre convexe est tout entier du même côté du plan d'une de ses faces, quelle qu'elle soit.

Pour mieux faire comprendre la nature des objets convexes et leurs propriétés, nous allons utiliser un exemple, illustré par la Figure VI.C. En effet, s'il est facile d'expliquer la convexité pour des polygones (les polygones convexes n'ont pas d'angles *rentrants*), il est parfois difficile de repérer un polyèdre concave, et ce pourrait être une source d'erreur dans la suite, d'où la nécessité d'être clair.

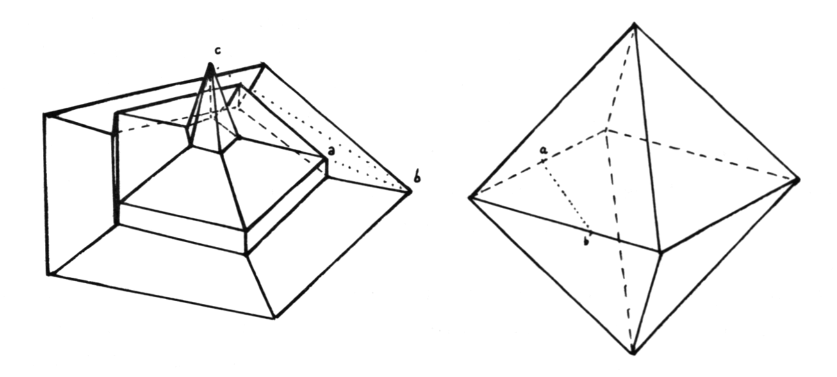


Figure VI.C

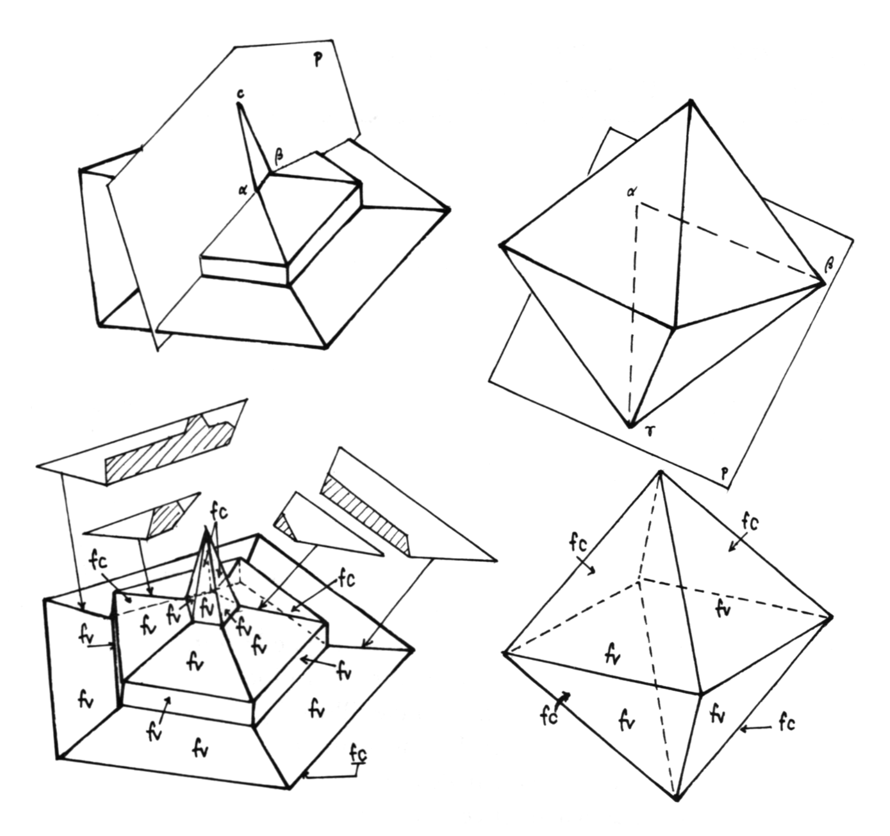


Figure VI.C, suite.

Dans cette figure, j'ai choisi pour objet de mes explications deux polyèdres. Celui de droite à huit faces et six sommets : c'est un octaèdre. Chacune de ses faces est identique aux autres, et est un triangle équilatéral. On le dit pour cela régulier (c'est aussi le cas du cube). L'autre, à base pentagonale, sera appelé dans la suite PCC, c'est-à-dire Polyèdre ConCave. Car il est bien concave. Observez, en haut, le segment [*ab*] ou le segment [*bc*]. Il est clair que ces segments sont en dehors du PCC. Si celui-ci était en bois, on pourrait tendre entre les points *b* et *c* une corde à linge : elle ne toucherait pas les faces.

Ce n'est pas le cas de l'octaèdre. En réfléchissant, vous verrez aisément qu'il n'est plus question ici de tendre une corde à linge, à moins de le faire creux. Ainsi le segment *ab* est entièrement inclus à l'intérieur, et il en serait de même de tout autre segment : au mieux serait-il dans une des faces, mais jamais à l'extérieur.

Nous pouvons donc affirmer que l'octaèdre, lui, est convexe.

Avant de vérifier que l'un possède les propriétés citées au début et l'autre pas, regardons la forme des faces de ces polyèdres.

Pour l'octaèdre, c'est tout simple ; j'ai déjà dit en effet que ses faces étaient triangulaires. Elles sont donc convexes.

Pour le PCC, il a (de haut en bas) : cinq faces triangulaires ; cinq faces en trapèze ; cinq faces rectangulaires ; encore cinq faces en trapèze ; une face pentagonale (la base). Ces faces sont donc toutes convexes. Nous voyons donc qu'un polyèdre peut avoir toutes ses faces convexes sans être convexe lui-même. Par contre, le polyèdre de la Figure VI.B, schéma *c*, a des faces non convexes comme AEIBF (sur cette face, l'angle EIF est *rentrant*). Cette remarque est à prendre en compte, comme nous le verrons sous peu.

Par contre, j'ai dit qu'un polyèdre convexe était tout entier du même côté du plan d'une quelconque de ses faces. Cette propriété est suffisante, en ce sens que les polyèdres concaves ne la vérifient jamais. Ainsi, sur la Figure VI.C, au milieu, nous voyons qu'une partie seulement du PCC se trouve en avant du plan P, plan de la face $\alpha\beta c$. Par contre, l'octaèdre est entièrement du même côté du plan P, plan de la face $\alpha\beta\gamma$. Le lecteur vérifiera qu'il en est de même des sept autres faces. Pour le PCC, c'est plutôt le contraire, puisque seule la base vérifie cette propriété, toutes les autres faces sont des contre-exemples comme $\alpha\beta c$.

Arrivé à ce stade du paragraphe, vous avez certainement compris ce qu'est un polyèdre convexe, mais sans doute vous demandez-vous pourquoi je fais tant de cas de cette propriété à l'apparence bien innocente.

Elle est en fait capitale pour la partie la plus difficile de ce chapitre, la gestion des parties cachées d'un polyèdre. Nous apprendrons en effet sous peu à ne faire dessiner à l'ordinateur que les faces visibles du polyèdre. Or, observons à présent le bas de la Figure VI.C. Nous constatons que, sur les vingt et une faces du PCC, douze sont visibles (fv), cinq sont cachées (fc), et surtout quatre sont partiellement visibles seulement. Ces quatre dernières faces ont été à nouveau représentées en *vue éclatée*, la partie hachurée représentant le morceau caché.

Même travail à droite sur l'octaèdre. Que voyons-nous ? Quatre faces visibles, quatre cachées mais *aucune qui ne soit que partiellement cachée*. Ce n'est pas là le fruit du hasard, mais bien celui de la convexité.

Dès lors, vous commencez à voir pourquoi j'ai tant insisté sur la définition par face des polyèdres d'une part, sur la convexité d'autre part... Grâce à cela, un seul paramètre nous permettra, pour une face donnée

d'un polyèdre convexe, de savoir si elle est visible ou non, suivant qu'il sera positif ou non.

Peu importe pour l'instant la valeur de ce paramètre, nous le verrons plus loin. L'essentiel, ici, est de bien comprendre l'intérêt de la convexité. Imaginez votre Amstrad dessinant l'octaèdre. Avant de tracer chaque face de la manière vue précédemment, il fait un test et trace *toute* la face si le résultat est positif, sinon il *passe à la suivante*. Au contraire, imaginez-le traçant le PCC. Que devra-t-il faire lorsqu'il aura à tracer l'une des faces qui n'est que partiellement visible ? Qu'il la trace entièrement ou qu'il la saute, le résultat sera mauvais. Il ne peut même pas tracer la partie visible en une seule fois, car celle-ci peut être formée de deux morceaux (voyez les faces de gauche), voire trois, ou bien plus sur des polyèdres concaves plus compliqués. On voit donc que chaque face devra alors être tracée point par point, et chaque point devra subir un test.

La gestion des faces cachées sera alors bien plus complexe et aussi bien plus longue. Quelques trucs facilitant le travail seront donnés au Chapitre VII, mais j'espère avoir convaincu le lecteur de la commodité relative des polyèdres convexes.

5. POLYEDRES A REPETITION CYLINDRIQUE

Souvenez-vous du cube de la Figure VI.B. Si vous avez cherché à le définir sous la forme d'une matrice par face pour tester le Programme VI.1, vous avez dû rentrer pour cela ce que nous appellerons des paramètres, c'est-à-dire des indications chiffrées ou des équivalents (notamment des *flags*(1)). Je vous propose de les compter. Tout d'abord, il y a huit sommets, chacun de trois coordonnées, soit vingt-quatre paramètres. Puis il y a six faces, chacune de quatre sommets, soit encore vingt-quatre paramètres. Ajoutons six flags pour l'orientation de chaque face, et nous arrivons à un total de cinquante-quatre paramètres, et c'est pourquoi vous avez dû vous amuser follement en le rentrant, si vous l'avez fait !

Les paragraphes suivants vont vous permettre de simplifier ce travail (après tout, l'ordinateur est fait pour cela...) dans un grand nombre de cas. Par exemple, celui-ci vous permettra d'obtenir le même cube avec six paramètres, soit... neuf fois moins !

(1) On appelle *flags* des indicateurs équivalant à un bit, c'est-à-dire ne prenant que deux valeurs, 0 et 1, ou parfois − 1 et 1.

Expliquons comment ce "miracle" est possible. Vous avez certainement déjà ouvert un parapluie. Ce commode ustensile, une fois grand ouvert, peut être sommairement décrit ainsi : autour d'un axe principal, que nous appellerons Oz pour simplifier, une baleine, qui est une sorte de ligne brisée faite en fil de fer, est fixée. Un certain nombre de baleines exactement semblables sont également fixées tout autour, de la même façon, à des angles augmentant régulièrement à partir de la baleine de départ. Celle-ci n'est d'ailleurs distinguée qu'arbitrairement car, si le parapluie a par exemple six baleines, une rotation autour de l'axe Oz de 360/6 = 60° ne changera rien à l'aspect du parapluie.

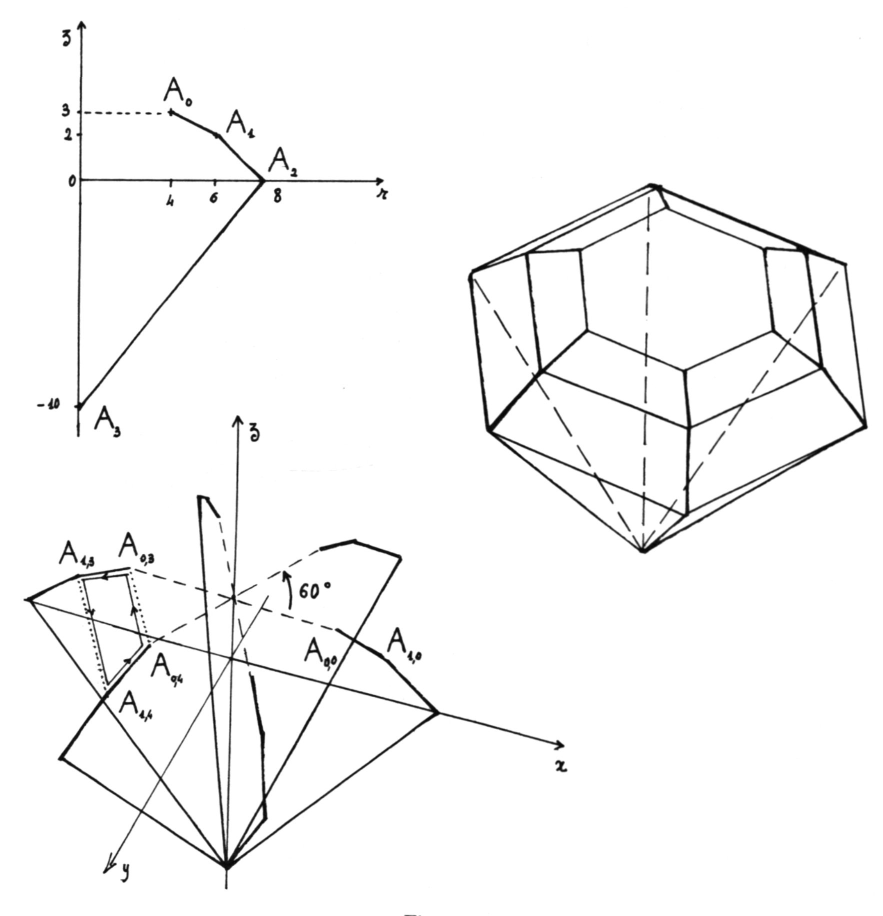


Figure VI.D

Nous allons à présent nous transformer en fabricants de parapluies, mais de parapluies un peu bizarres. Observez la Figure VI.D. En haut, à gauche, j'ai dessiné une baleine : c'est une ligne brisée $A_0 A_1 A_2 A_3$. L'axe O*z* est indiqué, ainsi qu'un axe O*r* de coordonnées cylindriques (voir Figure II.A). Cette baleine est ici attachée par le bas à O*z*, mais ce n'est nullement nécessaire.

Puis cette baleine va être répétée six fois autour de l'axe O*z*, l'axe O*r* (fictif) tournant à chaque fois de 60° autour de O*z*, sa position initiale coïncidant avec O*x*. Chaque point est alors répété six fois, A_0 devenant ainsi $A_{0,0}$, $A_{0,1}$, $A_{0,2}$, etc., à l'exception toutefois de A_3 dont toutes les positions successives sont confondues, car ce point est placé sur O*z*. On obtient alors un squelette représenté en bas de la figure.

Mais sur tout parapluie qui se respecte, on tend une toile qui en assure la cohésion tout en donnant sa protection à celui qui s'abrite. Nous allons faire de même. Deux segments successifs, comme $A_{0,3} A_{1,3}$ et $A_{0,4} A_{1,4}$, vont constituer une face. Toutes les faces ainsi constituées, nous obtiendrons le polyèdre dessiné à droite, qui d'ailleurs évoque plus un diamant de bijouterie qu'un parapluie.

Notez que ce diamant contient 19 faces, dont treize non triangulaires. Pour le constituer directement, il aurait fallu quelque $3 \times 20 + 6 + 12 \times 4 + 6 \times 3 + 13 = 145$ paramètres, alors qu'ici il suffit des deux coordonnées de chacun des quatre points de la *baleine*, et du nombre de répétitions à exécuter, soit neuf paramètres seulement... et d'un bon programme, que nous allons exposer à l'instant.

Il s'agit du Programme VI.2. Ce programme est destiné à être utilisé ajouté au Programme VI.1 (qui contient lui-même des programmes antérieurs, voir sa description), ainsi qu'avec les lignes de commencement que j'ai pris l'habitude de passer sous silence : mode, couleurs, etc., et ici un GOTO 3000 car c'est à cette ligne que tout commence.

La partie du programme donnée ici (lignes 3000 et suivantes) est donc simplement destinée à engendrer de manière simple le polyèdre, son utilisation ultérieure relevant du Programme VI.I déjà vu, ou de tout autre programme qu'il vous plaira d'inventer, pour vos propres utilisations.

Au vu de ce qui a été dit précédemment, et à l'aide de la Figure VI.D, les opérations à effectuer dans ce but peuvent paraître relativement simples. Il subsiste néanmoins quelques difficultés.

La plus importante est posée par la forme des faces. La plupart de ces faces sont tétragonales, c'est-à-dire à quatre sommets. Ainsi de la face $A_{0,3} A_{1,3} A_{1,4} A_{0,4}$ (tous les exemples qui suivent sont empruntés à la

Figure VI.D). Celle-ci, comme presque toutes, est formée de deux segments identiques, résultant d'un des segments de la baleine d'origine (le segment $A_0 A_1$ en l'occurrence), reliés entre eux par deux autres segments parallèles, soit au total quatre arêtes.

Mais l'on voit sur la figure des exceptions. Notamment les six faces les plus basses du polyèdre sont triangulaires car, le point A_3 appartenant à l'axe Oz, il est invariant par les rotations que l'on fait subir à notre *baleine* autour de cet axe, et par conséquent $A_{3,0} = A_{3,1} = A_{3,2} = \dots$

Pour résoudre ce premier problème, nous allons considérer que le point A n'est pas strictement sur l'axe Oz, mais qu'il en est tout près (r inférieur à $0,01$). Dès lors, les six faces inférieures sont comme les autres tétragonales, mais avec deux de leurs sommets si proches l'un de l'autre que l'œil, et même l'écran, ne saurait les distinguer. Mais si l'œil ne sait pas faire la différence, la machine la fera bien, elle, et cette subtile différence lui permettra d'appliquer à ces faces les mêmes automatismes qu'aux autres, aussi bien pour leur création que pour leur tracé ou pour leurs vecteurs normaux (voir le paragraphe à ce sujet).

Plus ardu est le problème posé par la face supérieure. En effet, le point A_0 n'étant pas sur Oz, notre baleine n'est pas attachée par les deux bouts à cet axe. Dès lors, à l'inverse du problème précédent, cet éloignement du premier point engendre une face supplémentaire, la face supérieure, qui a autant de sommets qu'il y a de *répétitions*, soit six dans le cas de figure.

Cette (ou ces) face(s) supplémentaire(s) (car le dernier point peut lui aussi ne pas être sur Oz, dans d'autres cas de figure) peut être séparée des autres ; elle a en général plus de sommets, son vecteur normal est évident (troisième vecteur du repère, ou son opposé pour la face inférieure éventuelle), etc.

Toutefois, pour des raisons de simplicité, ce n'est pas la solution choisie dans le Programme VI.2 qui incorpore cette face à la matrice globale des faces *a*.

PROGRAMME VI.2

```
3000 '================================== Polyedre a repetition cylindrique
3010 CLS:INPUT "Nombre d'aretes dans un plan diametral ";k
3020 CLS:INPUT "Nombre de repetitions ";n
3030 f=n*k:DEG:w=360/n
3040 CLS:INPUT "Echelle d'affinite de x par rapport a z";Ex
3050 PRINT:INPUT "Echelle d'affinite de y par rapport a z";Ey
3060 DIM b(k,1)
3070 CLS:PRINT"ATTENTION!""Donnez les points dans un plan diametral"
```

```
3080 PRINT"en les reperant avec 2 coordonnees (r,z)"
3090 PRINT"z allant en DECROISSANT.":FOR T=0 TO 3000:NEXT
3100 CLS:PEN 3:PRINT"1ere arete":PEN 1
3110 PRINT"1er point"
3120 INPUT "r=",b(0,0)
3130 INPUT "z=",b(0,1)
3140 PRINT"2eme point"
3150 INPUT "r=",b(1,0)
3160 INPUT "z=",b(1,1)
3170 CLS
3180 FOR y=2 TO k
3190 PEN 3:PRINT y"eme arete":PEN 1
3200 PRINT:PRINT"Le premier point est le deuxieme de ""l'arete precedente."
3210 PRINT"2eme point"
3220 INPUT "r=",b(y,0)
3230 INPUT "z=",b(y,1)
3240 NEXT
3250 ff=f:smax=3
3260 IF b(0,0)=0 THEN b(0,0)=0.01
3270 IF b(k,0)=0 THEN b(k,0)=0.01
3280 IF b(0,0)>0.01 THEN f=f+1:smax=n-1
3290 IF b(k,0)>0.01 THEN f=f+1:smax=n-1
3300 IF smax<3 THEN smax =3
3310 DIM a(f-1,smax,2)
3320 FOR t=0 TO ff-1
3330 i=INT(t/n):p=t-i*n
3340 c=Ex*COS(p*w):s=Ey*SIN(p*w)
3350 a(t,0,2)=b(i,1)
3360 a(t,0,0)=c*b(i,0)
3370 a(t,0,1)=s*b(i,0)
3380 a(t,1,2)=b(i+1,1)
3390 a(t,1,0)=c*b(i+1,0)
3400 a(t,1,1)=s*b(i+1,0)
3410 c=Ex*COS((p-1)*w):s=Ey*SIN((p-1)*w)
3420 a(t,2,2)=b(i+1,1)
3430 a(t,2,0)=c*b(i+1,0)
3440 a(t,2,1)=s*b(i+1,0)
3450 a(t,3,2)=b(i,1)
3460 a(t,3,0)=c*b(i,0)
3470 a(t,3,1)=s*b(i,0)
3480 NEXT
3490 DIM AA(f-1):FOR t=0 TO f-1:AA(t)=4:NEXT
3500 IF b(0,0)<=0.01 GOTO 3590
3510 IF b(k,0)<=0.01 THEN t=f-1 ELSE t=f-2
3520 AA(t)=n
3530 FOR p=1 TO n
3540 a(t,p-1,2)=b(0,1)
3550 c=Ex*COS((p-1)*w):s=Ey*SIN((p-1)*w)
3560 a(t,p-1,0)=c*b(0,0)
3570 a(t,p-1,1)=s*b(0,0)
3580 NEXT
3590 IF b(k,0)<=0.01 GOTO 3670
3600 AA(f-1)=n
3610 FOR p=1 TO n
3620 a(f-1,p-1,2)=b(k,1)
3630 c=Ex*COS((p-1)*w):s=Ey*SIN((p-1)*w)
3640 a(f-1,p-1,0)=c*b(k,0)
3650 a(f-1,p-1,1)=s*b(k,0)
3660 NEXT
3670 GOTO 1300
```

Passons dès à présent à ce programme. Il débute par quatre INPUT, qui engendrent quatre nombres, *n*, *k*, E*x* et E*y*. “*k*” est le nombre d'arêtes de la baleine. Ce nombre permet de dimensionner correctement la matrice *b* des points de la baleine. “*n*” est le nombre de répétitions, six dans le cas de

figure (où k valait trois, soit dit en passant). "Ex" et "Ey" sont des échelles d'affinité [1]. L'introduction de ces nombres permet d'étendre encore les possibilités de variation de ces polyèdres, en les *écrasant* un peu perpendiculairement à Ox (Ex inférieur à 1), ou à Oy, ou au contraire en les *dilatant* (Ex supérieur à 1)... Mais, en général, nous leur laisserons la valeur normale 1.

On calcule ensuite (ligne 3030) le nombre f de faces du polyèdre, *faces supérieures et inférieures non comptées*, soit $n * k$, ainsi que l'angle de rotation élémentaire $w = 360/n$ (en degrés), soit 60° sur l'exemple.

La matrice b, je l'ai déjà dit, contient les $k+1$ points de la baleine, qui ont chacun deux coordonnées (puisque la baleine est plane).

Ensuite, une série de INPUT vous demandent, dans l'ordre, les coordonnées des susdits points. L'ordre en question est important, sinon vous risquez d'obtenir n'importe quoi. L'ordre suggéré est celui où z décroît, mais il n'est nullement obligatoire.

Pour un résultat optimal au cours de l'exécution du programme, je vous conseille de tracer votre baleine d'abord sur le papier, comme je l'ai fait en haut de la Figure VI.D, afin d'éviter des erreurs ou des incohérences (du type : annoncer quatre arêtes et n'en faire que trois), qui vous feraient... accuser injustement mon programme !

Arrivé à la ligne 3250, le programme a avalé votre baleine avec ses arêtes... Reste à la digérer. Et tout d'abord, il s'attaque aux problèmes dont nous avons parlé. Notez que ce problème ne concerne que les faces inférieures et supérieures du polyèdre, et plus précisément les premier et dernier points de la baleine, dont il s'agit de savoir s'ils sont ou non sur l'axe Oz.

Si l'un y est (cas du point A_3), alors on le décale très légèrement (lignes 3260, 3261), comme décrit précédemment (la différence restera imperceptible). Dans le cas contraire, il faut créer une face supplémentaire, qui, de plus, aura n sommets. Or s_{max} joue ici le rôle de $S-1$, où S est le nombre maximal de sommets par face. En général, il y a $3+1$ sommets par face (faces normales tétragonales), d'où $s_{max} = 3$. Mais s'il faut rajouter une ou deux faces (supérieure et inférieure), comme à cause du point A_0, s_{max} augmente et vaut $n-1$ (sauf si $n = 3$, d'où la ligne 3300). On obtient donc ainsi les lignes 3260 à 3290. Il ne reste plus qu'à dimensionner la matrice des faces a au nombre exact de faces, ayant au plus $s_{max}+1$ sommets de $2+1=3$ coordonnées chacun (ligne 3310).

(1) Une affinité est une application linéaire de matrice type : $\begin{pmatrix} k & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Reste à présent à engendrer cette matrice des faces, c'est-à-dire à la remplir.

Les faces sont numérotées de 0 à $ff - 1$ (avec $ff = n * k$) par l'indice t. L'indice i, variant de 0 à $k - 1$, est le premier indice des deux points les plus hauts de la face (je parle ici des faces normales tétragonales comme $A_{0,3} A_{1,3} A_{1,4} A_{0,4}$). L'indice p, variant de 1 à n, est le deuxième indice des points de la face qui ont le plus gros deuxième indice. L'explication de ces deux indices par point est illustrée par la Figure VI.D. De ce fait, la face t est celle qui contient les quatre sommets d'indices (i,p) ; $(i+1,p)$; $(i+1,p-1)$; $(i,p-1)$, et dans cet ordre.

Les trois indices sont liés par les formules de la ligne 3330. Sachant que le point d'indice (i,p) se déduit du point n° i de la baleine par une rotation autour de Oz d'angle $p * w$, et au vu des formules de rotation données au Chapitre II, tout à présent doit être à peu près clair jusqu'à la ligne 3480. En 3490, on dimensionne la matrice AA (qui donne le nombre de sommets contenus dans chaque face). Presque toutes les faces ont quatre sommets, les faces triangulaires étant considérées comme tétragonales.

Reste à engendrer, s'il y a lieu, la face inférieure et la face supérieure. Pour ces faces, AA (t) vaut n (n sommets), et les sommets sont toutes les images d'un même point de départ de la baleine, soit le premier A_0 (c'est le cas de la figure), soit le dernier pour une éventuelle face inférieure.

Le programme s'oriente ensuite vers les routines de calcul d'échelle, de dessin, etc., que nous avons déjà vues au Paragraphe 3.

Je tiens à préciser que ce programme est très loin d'être optimal. Je n'ai néanmoins pas cherché à l'améliorer pour ne pas trop le compliquer (voyez déjà la longueur des explications). Mes lecteurs doués pourront peut-être simplifier l'enregistrement de la baleine à l'aide du plot baladeur, ou encore, notant que chaque arête est tracée deux fois, trouver le moyen de résoudre ce petit inconvénient. Comment ? *(Question VI.a.)* De plus, il est possible, pour gagner de la place mémoire, de séparer les faces tétragonales dans une matrice à part, etc.

Je crois toutefois que le programme, même non optimisé, vous permettra de créer sans difficulté des polyèdres très variés, comme celui de la Figure VI.D (les coordonnées à porter y sont indiquées), comme la grosse sphère de la Figure VI.F, comme le tétraèdre et le PCC de la Figure VI.C, comme le cube (rentrer une seule arête verticale et $n = 4$) ou toute autre des multiples formes que votre imagination ou le hasard vous suggéreront... Même un parapluie, pourquoi pas !

6. POLYEDRE CONVEXE D'UNE FAMILLE DE POINTS

Les mathématiques fourmillent de ces petits théorèmes qui n'ont en eux-mêmes guère d'utilité, mais auxquels on peut parfois trouver un usage commode. Celui, assez simple, qui suit peut servir comme on le verra.

Soit $A_0 A_1 A_2 \ldots A_{n-1}$, une famille de points de l'espace[1]. Il existe une infinité de polyèdres qui contiennent cette famille (c'est-à-dire qui contiennent tous les points de la famille), et même une infinité de polyèdres convexes. Mais, parmi tous ces polyèdres convexes contenant la famille, il en existe un, et un seul, qui soit contenu dans tous les autres : c'est le polyèdre convexe minimal (PCM) de la famille.

Cette propriété vous paraît peut-être évidente. Notez que toutefois elle est fausse pour les polyèdres concaves.

Ce PCM a de nombreuses propriétés. Outre la convexité résultant de sa définition, on vérifie en particulier que tous les sommets $S_0 S_1 \ldots S_{S-1}$ de ce polyèdre font partie de la famille (A_i) ; la réciproque est fausse, un certain nombre de points de la famille pouvant très bien se trouver à l'intérieur du PCM, ou bien sur une face ou une arête (c'est le cas par exemple si trois des points de la famille sont alignés). Mais une chose est sûre : le nombre s de sommets du PCM est inférieur ou égal au nombre n de points de la famille.

Le PCM d'une famille de points peut être entièrement engendré par ordinateur. L'avantage est grand, car cela évite d'avoir à donner les paramètres d'orientation des faces et de leur nature ; vous donnez simplement les triplets de coordonnées de chaque sommet du polyèdre que vous voulez créer, et l'ordinateur fait le reste. L'*inconvénient* résultant est que le polyèdre obtenu est convexe ; si cela est un avantage du point de vue des parties cachées, comme je l'ai déjà dit, cela n'en est pas moins limitatif, et les polyèdres concaves ne peuvent être conçus par la méthode exposée ci-dessous. D'une façon générale, les polyèdres concaves sont pénibles à engendrer, pénibles à dessiner, pénibles à utiliser... mais nous aurons l'occasion d'en reparler dans un autre paragraphe.

Comment donc engendrer ce polyèdre ? Cela n'est pas vraiment simple, car deux points quelconques n'appartiennent pas forcément à la même arête, ni même à une arête, ni trois points à la même face.

(1) On appelle famille de points un ensemble de points numérotés dans un ordre plus ou moins arbitraire.

Pour engendrer un PCM, il faut avoir recours à un *algorithme*. Ce mot pompeux désigne tout simplement le fait de répéter une même opération sur un ensemble d'objets, jusqu'à épuisement du stock. Ici les objets sont les points et les faces qu'ils déterminent, jusqu'au moment où il n'y a plus de face à créer.

Un algorithme est précédé d'une *initialisation*, ou *amorce*, et est suivi d'un *arrêt*, en général basé sur une condition.

Tout cela sera plus clair sur un exemple. Nous voulons engendrer notre PCM en créant à partir de n points, de coordonnées connues, les faces de leur PCM.

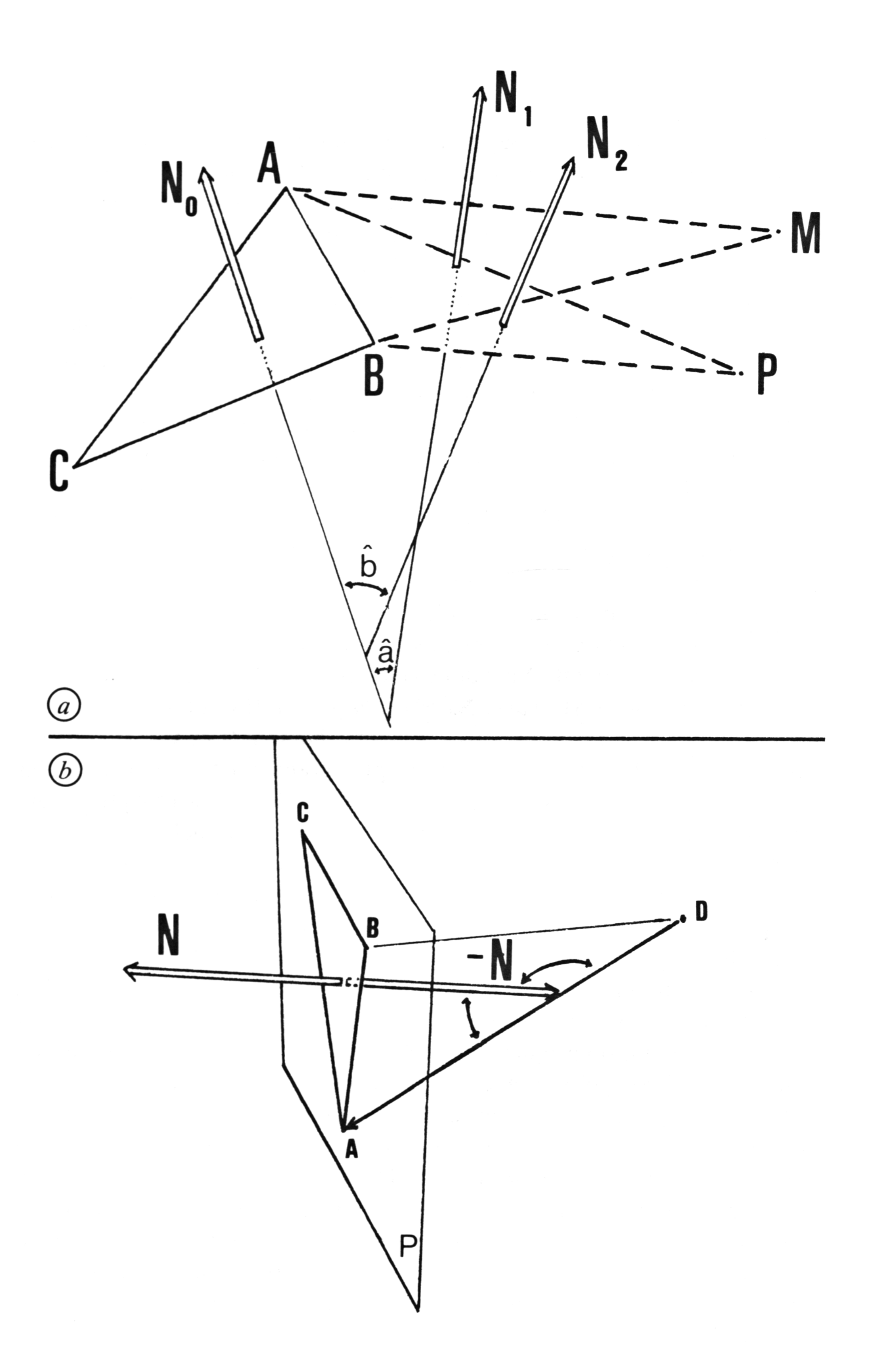
L'algorithme, on le verra dans un instant, permet d'obtenir une nouvelle face à partir d'une ancienne face. Pour pouvoir l'utiliser il faut donc connaître au moins une face, d'où la nécessité d'une initialisation consistant à rechercher une première face.

Cette initialisation se déroule ainsi : on commence par rechercher les deux points de la famille qui ont la plus faible coordonnée x, par exemple. Appelons ces points A et B. Nous admettrons que le segment AB est obligatoirement une des arêtes du PCM. Dès lors, il existe un point C au moins tel que ABC forme une face. Nous ne nous intéressons en effet ici qu'aux faces triangulaires, toute face plus grande pouvant toujours être considérée comme la réunion de plusieurs faces triangulaires. Pour obtenir le point C, on se réfère à la définition de la convexité ; le PCM doit être tout entier du même côté de la face ABC. Dans les faits, cela se traduit ainsi : si **N** est le vecteur normal à la face ABC (nous verrons tout à l'heure comment l'obtenir), les produits scalaires **N** · **MA** doivent tous être positifs ou nuls. On prend donc un point C, on teste tous les produits scalaires après avoir calculé le vecteur normal à la face supposée ABC ; si l'un d'entre eux est négatif, le point ne convient pas, il faut choisir un autre C (le mieux est de les essayer dans l'ordre x croissant), sinon on a enfin obtenu notre première face.

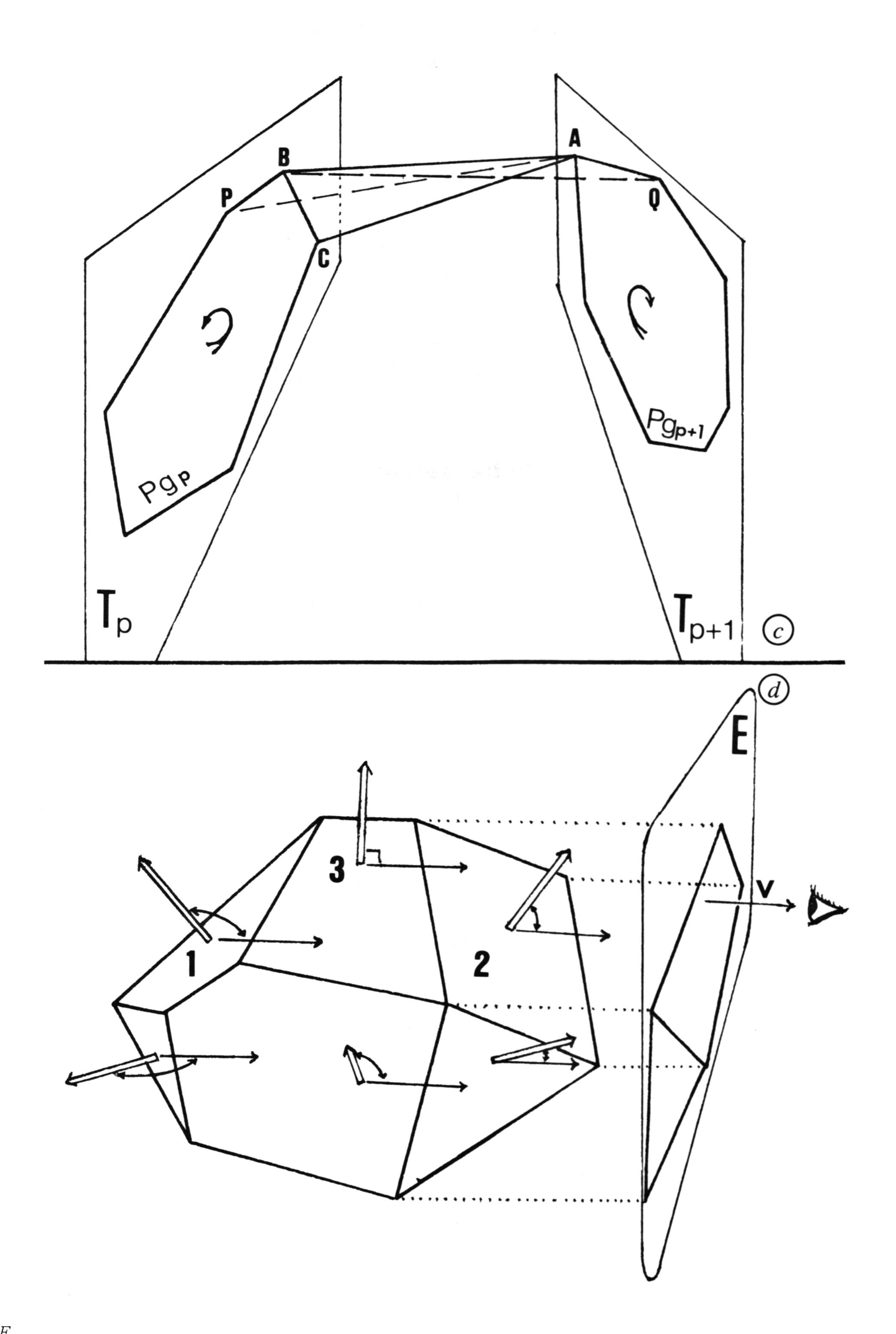
C'est alors que l'algorithme entre en action. En effet, on pourrait continuer à déterminer toutes les faces comme la première, mais ce serait trop long car, en fait, c'est un peu une recherche "au petit bonheur" qui entraîne d'innombrables tests à chaque essai... et il peut y avoir beaucoup d'essais.

Or il y a plus simple. Supposons que nous ayons une face ABC, de vecteur normal $\mathbf{N}_0$ (dont nous noterons n_0 la norme, car il n'est pas nécessaire qu'il soit normé). L'arête AB appartient nécessairement à une autre face ABD. Il faut donc trouver le point D. Pour cela, admettons que nous hésitions entre deux points M et P (voir Figure VI.E, schéma *a*). Soit $\mathbf{N_1}$ et $\mathbf{N_2}$ les vecteurs perpendiculaires aux triangles ABM et ABP respectivement, orientés positivement (explication plus loin), et de plus *normés* obligatoirement. Soit aussi ps_1 et ps_2 les produits scalaires de ces vecteurs avec $\mathbf{N_0}$. On peut voir sur la figure que le point P est *en dessous* de M ; c'est donc ce dernier point qui est le bon. Si $\hat{a}$ et $\hat{b}$ sont les angles séparant les vecteurs en question, on constate également sur la Figure VI.E.*a* que $\hat{a} < \hat{b}$. Or, cela est général ; nous admettrons que le point D recherché est tel que l'angle séparant le vecteur orthogonal à ABD de $\mathbf{N_0}$ est le plus petit possible ; c'est par conséquent celui qui a le cosinus le plus grand. Or, le produit scalaire entre $\mathbf{N_0}$ et $\mathbf{N_1}$, par exemple, vu que $\mathbf{N_1}$ est normé, vaut (voir Chapitre II) : $ps_1 = n_0 \cos(\hat{a})$. De même, $ps_2 = n_0 \cos(\hat{b})$. D'où, n_0 étant positif (norme), $\hat{a} < \hat{b}$ est équivalent de $ps_1 > ps_2$. Le *bon* point D est donc celui qui est tel que *le produit scalaire du vecteur normal à* ABD *avec* $\mathbf{N_0}$ *est le plus grand.* Il suffit donc de calculer ces produits scalaires pour tous les points, hors A, B et C, et de choisir comme point D celui dont résulte le plus grand produit. On aura ainsi obtenu une nouvelle face ABD. Il n'y a plus alors qu'à rebaptiser cette nouvelle face ABC, et à recommencer l'algorithme au début.

Notez que je n'ai cessé de parler *du* vecteur normal à une face du polyèdre. Or, on sait qu'il existe, à norme donnée, *deux* vecteurs perpendiculaires au plan P d'une face ABC : **N** et son opposé $-\mathbf{N}$. Lequel est le bon ? (Voir Figure VI.E, schéma *b*.) Pour un polyèdre, on décide arbitrairement que le bon est celui qui *sort* du polyèdre, c'est-à-dire qui est orienté de l'intérieur à l'extérieur de la face. Le polyèdre étant convexe, si l'on se donne un point quelconque D du polyèdre n'appartenant pas à la face ABC, **N** doit pointer du côté opposé à D, et donc les vecteurs **N** et **DA** sont séparés par un angle aigu, donc leur produit scalaire est positif, alors que **DA** et $-\mathbf{N}$ sont séparés par un angle obtus, d'où un produit scalaire négatif. Si le produit scalaire est nul, D est dans P, il faut donc choisir un autre point de test. Ainsi, pour trouver **N**, on procède de la sorte : on calcule $\mathbf{V} = \mathbf{AB} \wedge \mathbf{AC}$ (par exemple) ; on prend un point D de la famille, et on calcule $ps = \mathbf{V} \cdot \mathbf{DA}$; si ps est nul, on recommence avec un autre point D ; si ps est strictement positif, alors $\mathbf{N} = \mathbf{V}$, sinon $\mathbf{N} = -\mathbf{V}$. C.Q.F.D.

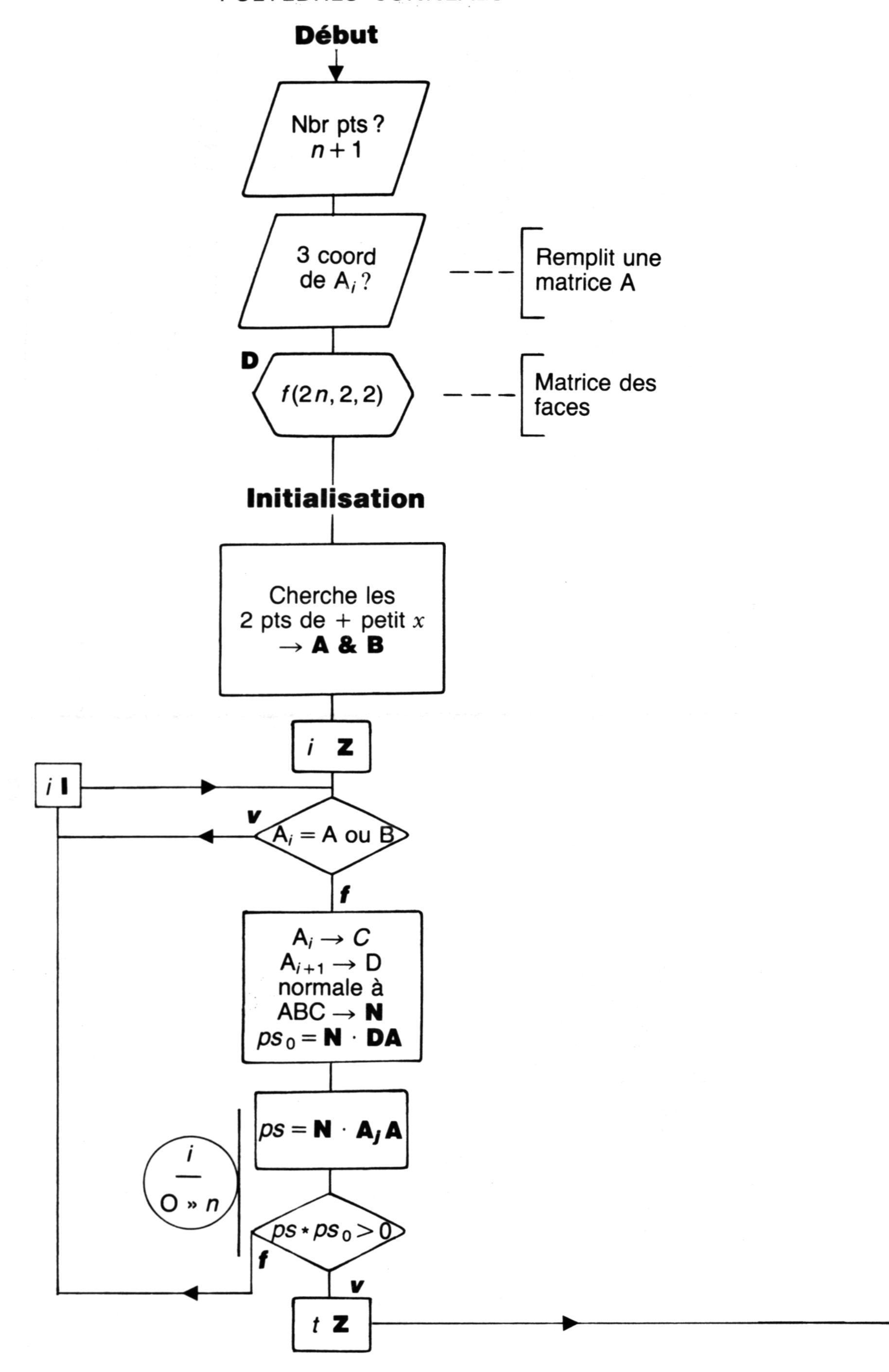


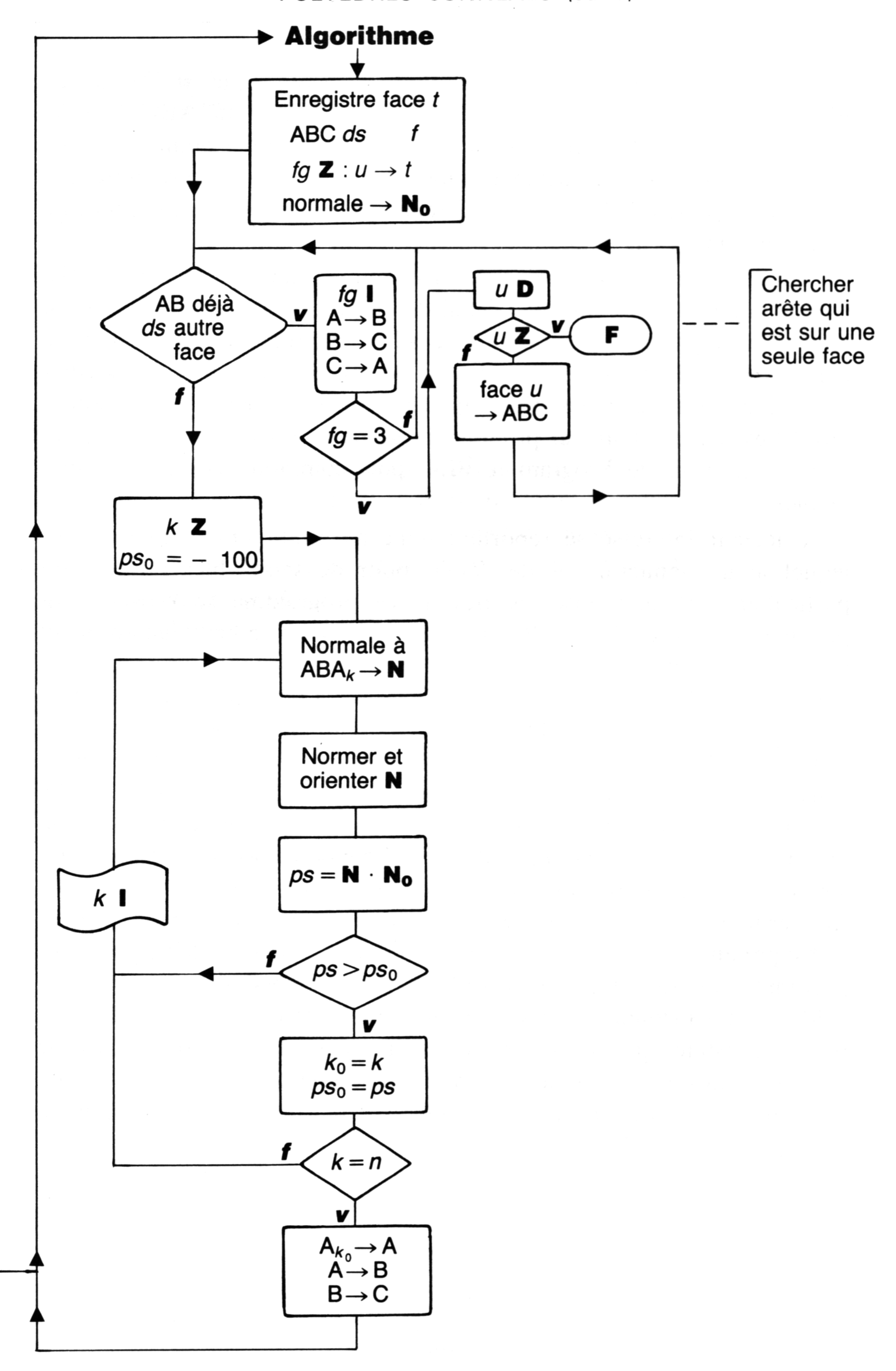
Figure



B
A
P
Q
C
Pgp
Pgp+1
Tp
Tp+1
c
d
E
3
1
2
V

POLYÈDRES CONNEXES





Algorithme
Enregistre face t
ABC ds f
fg Z : u → t
normale → N0
AB déjà ds autre face
v
f
fg I
A → B
B → C
C → A
fg = 3
f
v
u D
u Z
v
F
f
face u
→ ABC
Chercher arête qui est sur une seule face
k Z
ps0 = − 100
Normale à ABAk → N
Normer et orienter N
ps = N · N0
k I
f
ps > ps0
v
k0 = k
ps0 = ps
f
k = n
v
Ak0 → A
A → B
B → C

L'algorithme, naturellement, ne doit s'exécuter que si l'arête AB n'appartient encore à aucune des faces trouvées, autre que ABC. Dans le cas contraire, il faut exécuter un changement de nom des points, voire de face. Lorsque enfin il devient impossible de trouver une arête qui n'appartienne déjà à deux faces différentes, alors c'est l'arrêt : le polyèdre a été totalement engendré.

Tout cela se trouve réuni dans le Programme VI.3 dont seul, toutefois, l'organigramme est donné. Le détail de ce programme ressemblerait en effet beaucoup à celui du Programme VI.4 ; en outre, un tel programme me paraît trop lent pour être utilisé en BASIC. C'est pourquoi le lecteur initié à d'autres langages plus rapides pourra, je pense sans grandes difficultés, exécuter un tel programme grâce à son organigramme. Les autres utiliseront le Programme VI.4, plus commode et de buts semblables.

Le lecteur intéressé se reportera donc au susdit organigramme, sur lequel je ne donnerai pas de détails pour de semblables raisons. En particulier, de nombreuses parties de ce programme se retrouveront presque inchangées dans le Programme VI.4, qui sera expliqué en détail dans le paragraphe suivant.

Un seul problème mérite d'être expliqué dès à présent. D'ailleurs, il ne sera pas repris ensuite, car il se reposera dans les mêmes termes. Avant de commencer à engendrer les faces, il faut *dimensionner* la matrice a qui les conserve. Or, pour cela, il faudrait connaître le nombre de faces du PCM à l'avance, en ne connaissant guère, en fait, que le nombre de ses sommets, et encore, car les n points initiaux ne sont pas forcément tous des sommets... C'est impossible. Cela peut toutefois s'arranger, si l'on peut trouver une *majoration* de f, c'est-à-dire un nombre F dont on est sûr qu'il soit supérieur à f. Or, on dispose de la formule suivante, valable pour tout polyèdre convexe ; s'il a f faces, s sommets et a arêtes, alors $f+s=a+2$.

Mais a est inconnu, s aussi (on sait seulement que $s \leqslant n$). Mais vous pouvez constater que, si chaque face est triangulaire, il y a trois arêtes par face et, d'autre part, chaque arête appartient à deux faces. Donc $2a=3f$, et il en résulte que $f=2(s-2)$. Donc $F=2(n-2)$ convient comme majoration. C.Q.F.D.

7. POLYEDRES "ETAGES"

Nous allons à présent nous attaquer à un problème semblable au précédent, dans ses termes et dans son application, mais sensiblement différent dans son utilisation et dans ses résultats.

Le nouveau système qui va suivre est basé sur une observation du précédent. Pour la fabrication du PCM, il faut d'abord donner les points à utiliser, avec leurs trois coordonnées, ce qui n'est guère commode.

Il est infiniment plus simple d'utiliser le *plot baladeur* que de donner des ribambelles de nombres. Mais voilà, le plot ne peut être utilisé que dans un plan, et tous les sommets d'un polyèdre ne sauraient être contenus dans un plan. N'importe, ils sont contenus dans *plusieurs* plans.

En effet, pour tout polyèdre, il est toujours possible de trouver une famille de plans T_i, tels que tout sommet du polyèdre soit contenu dans l'un des T_i, et de plus, tous parallèles entre eux et parallèles à un plan fixe donné arbitrairement, par exemple le plan xOy. Il suffit pour cela d'imaginer le polyèdre découpé en tranches comme un pain.

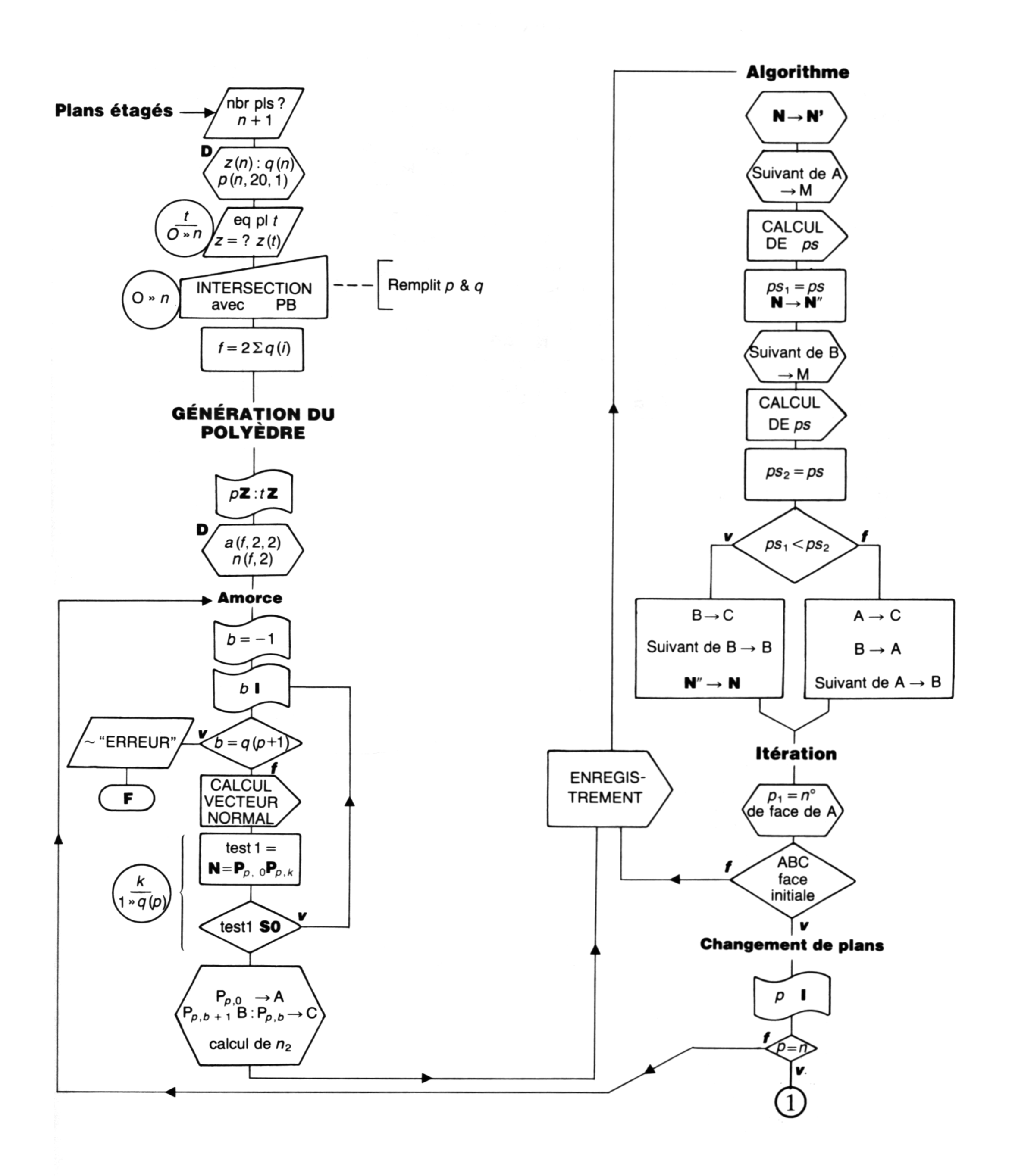
Or, l'intersection d'un plan avec un polyèdre est un polygone[1]. Dès lors, supposons qu'une famille de plans T_i, d'équation $z = z(i)$ où $z(0)$, etc. sont des constantes données, étant déterminée, nous donnons dans chaque plan T_i, à l'aide du plot baladeur, un polygone de $q(i) + 1$ sommets notés $P(i, 0)$, $P(i, 1)$, ..., $P(i, q(i))$. Puis, nous pouvons relier ces polygones en constituant des faces, le résultat étant un polyèdre en quelque sorte constitué par étages, comme une maison, que l'on a aussi l'habitude de décrire par des plans parallèles ; en effet, un architecte donne bien, à l'ingénieur chargé de construire la maison, des *plans*, c'est-à-dire des sections successives de la maison, étage par étage, soit autant de plans (parallèles ici à l'horizontale) qu'il y a d'étages. C'est pourquoi les polyèdres que nous constituerons de la même façon seront appelés *polyèdres étagés* (PE). L'architecte donne aussi la hauteur de chaque étage, ce qui équivaut à donner sa hauteur par rapport au sol $z(i)$. Que fait l'ingénieur ? Il relie les étages entre eux par des murs, jusqu'à obtention de la maison complète. Ici, vous êtes l'architecte, vous donnez les plans et les hauteurs. L'ordinateur est l'ingénieur, il construit le polyèdre en reliant chaque section à la suivante.

(1) ou plusieurs éventuellement. Mais n'épiloguons pas...

En principe, *tout* polyèdre peut être ainsi obtenu. Il suffit pour cela de le découper correctement. Le résultat sera légèrement distinct du polyèdre de départ, car il y aura plus de faces et plus de sommets, les sommets des polygones de chaque section n'étant pas forcément tous des sommets du polyèdre de départ. Ainsi, une seule face du polyèdre initial pourra se retrouver découpée en morceaux (coplanaires évidemment) enregistrés sous la forme de faces distinctes (encore que cet inconvénient puisse en réalité être résolu ; comment, à votre avis ?). *(Question VI.b.)* Mais la *forme* générale, qui est la plus importante, subsistera bien.

En fait, pour des raisons techniques qui vont bientôt apparaître, il faut que les polygones des sections soient convexes (on pourrait choisir des concaves, mais il faudrait *aider* un peu l'appareil, comme on le verra plus loin). Cela limite le nombre de polyèdres possibles, mais les polyèdres obtenus ne seront pas nécessairement convexes. On les dira *convexes par tranches* (CPT).

Pourquoi donc *convexes par tranches* ? C'est que, au cours de la génération du polyèdre, nous ne considérerons à la fois que deux plans T_i, le plan T_p et son successeur T_{p+1} (voir Figure VI.E, schéma *c*). Les autres plans seront (momentanément) oubliés. Dès lors, si l'on considère les deux polygones Pg_p et Pg_{p+1}, intersections du polyèdre à créer et des deux plans, il peuvent être considérés, à condition qu'ils soient convexes, comme deux des faces d'un polyèdre imaginaire PCX_p, compris entre les deux plans, convexe, et admettant comme sommets les $q(p) + 1$ sommets de Pg_p, et les $q(p+1) + 1$ sommets de Pg_{p+1}. Ce PCX_p est, en fait, on l'aura compris, le PCM de ces points du Pg_p et du Pg_{p+1} réunis, mais considérés seuls dans l'espace. Quant au vrai polyèdre, il sera en quelque sorte la réunion de tous les PCX_i superposés, tout comme une maison est la réunion de tous ses étages superposés.


Plans étagés
nbr pls ? n + 1
D
z(n) : q(n) p(n, 20, 1)
t / O » n
eq pl t z = ? z(t)
O » n
INTERSECTION avec PB
Remplit p & q
f = 2 Σ q(i)
GÉNÉRATION DU POLYÈDRE
pZ : tZ
D
a(f, 2, 2) n(f, 2)
Amorce
b = −1
b I
~ "ERREUR"
v
b = q(p+1)
f
F
CALCUL VECTEUR NORMAL
test 1 = N = P p,0 P p,k
k / 1 » q(p)
test1 SO
v
P p,0 → A
P p,b+1 B : P p,b → C
calcul de n2
Algorithme
N → N'
Suivant de A → M
CALCUL DE ps
ps1 = ps N → N"
Suivant de B → M
CALCUL DE ps
ps2 = ps
v
ps1 < ps2
f
B → C
Suivant de B → B
N" → N
A → C
B → A
Suivant de A → B
Itération
ENREGISTREMENT
p1 = n° de face de A
f
ABC face initiale
v
Changement de plans
p I
f
p = n
v.
1

CALCUL DU VECTEUR NORMAL

$P' = p + 1$
$b' = b + 1$
$\ell = 1$
$\mathbf{V} = \mathbf{P}_{p',b'}\mathbf{P}_{p,o} \wedge \mathbf{P}_{p'b}\mathbf{P}_{pb'}$

test 2 = $\mathbf{V} \cdot \mathbf{P}_{p',b'}\mathbf{P}_{p',b'+\ell}$

test 2 **Z** — v → ℓ **I** ; f ↓

test 2 **SO** — v → **N** = − **V** ; f → **N** = **V**

R

ENREGISTREMENT

A → a (t, 0, ...)
B → a (t, 1, ...)
C → a (t, 2, ...)
N → n (t, ...)

t **I**

R

CALCUL DE POINTS

N = **AB** ∧ **AM**
test 3 = **N** · **AC**

test 3 **Z** — v → c = n° de C ; f ↓

c **I**

test 3 **SO** — v ; f

test 4
N · **AP**

test 4 **Z** — v ; f

test 4 **SO** — v → **N** csg ; f

Normes **N**
ps = **N** . **N**′

R

①

FACES EXTRÊMES

$q_0 = q\ (0)$
$q_1 = q\ (n)$
$q = \max (q_0, q_1)$

D

fe (1, q, 2)

Chargement de fe (0, ...) avec le 1er polygone

Chargement de fe (1, ...) avec le dernier polygone

Polyedre fini

PROGRAMME VI.4

```
10 GOTO 5000
300 REM
305 IF flag=1 GOTO 470
490 FOR u=0 TO 2
522 FOR t=0 TO 1:FOR u=0 TO q
524 r=fe(t,u,0)*fe(t,u,0)+fe(t,u,1)*fe(t,u,1)+fe(t,u,2)*fe(t,u,2)
526 IF r>rmax THEN rmax=r
528 NEXT:NEXT
635 FOR u=0 TO 2
730 IF q1=1 GOTO 810
740 IF c2>=0 GOTO 810
750 FOR u=0 TO q0
760 x=fe(0,u,0):y=fe(0,u,1):z=fe(0,u,2):GOSUB 1200
770 IF u=0 THEN a=XX:b=YY:PLOT a,b:GOTO 790
780 DRAW XX,YY
790 NEXT u
800 DRAW a,b
810 IF q2=1 GOTO 2000
820 IF c2<=0 GOTO 2000
830 FOR u=0 TO q1
840 x=fe(1,u,0):y=fe(1,u,1):z=fe(1,u,2):GOSUB 1200
850 IF u=0 THEN a=XX:b=YY:PLOT a,b:GOTO 870
860 DRAW XX,YY
870 NEXT u
880 DRAW a,b
890 GOTO 2000
```

```
5000 '======================================================== Plans etages
5010 INPUT "Nombre de plans etages";n:n=n-1
5020 CLS:DIM z(n):DIM q(n)
5030 INK 2,6:PEN 2:INK 3,18
5040 PRINT TAB(7)"Ordonnee des plans etages"
5050 PEN 1
5060 PRINT:PRINT"(par ordre CROISSANT obligatoirement)":PRINT
5070 FOR t=0 TO n
5080 PRINT"Ordonnee du plan #"t+1;:INPUT z(t)
5090 NEXT
5100 CLS:DIM p(n,20,1)
5110 PRINT "A present,donnez les intersections avec" "les plans successifs
grace au plot""baladeur."
5120 PRINT"Il n'est pas necessaire que ces ""intersections soient fermees"
5130 PRINT   "Mais elles doivent etre convexes,"    "decrites dans le sens
trigonometrique,"
5140 PRINT "et leur premier point dans ce sens""angulaire doit preceder le
premier ""point de l'intersection precedente."
5150 PEN 2:LOCATE 1,25:PRINT"Appuyez sur la barre pour continuer":
WHILE INKEY(47)=-1:WEND
5160 ORIGIN 320,200
5170 FOR l=1 TO n+1
5180 CLS:TAG
5190 PLOT 0,-200,3:DRAW 0,180:PRINT"y";
5200 PLOT -320,0:DRAW 300,0:PRINT"x";
5210 TAGOFF
5220 y=3
5230 FOR p=0 TO l-2
5240 GOSUB 12000
5250 DRAW p(p,0,0),p(p,0,1)
5260 DRAW p(p,1,0),p(p,1,1)
5270 NEXT
5280 p=l-1
5290 y=1
5300 PEN 3:LOCATE 25,1:PRINT"Plan numero"l
5310 GOSUB 10000
5320 NEXT
5330 DEFINT f:f=0
5340 FOR l=0 TO n
5350 f=f+q(l)+1
5360 NEXT
5370 f=2*f
6000 '=====================================================================
6010 '====================== GENERATION DU POLYEDRE ======================
6020 '=====================================================================

6030 p=0:t=0
6040 DIM a(f-1,2,2):DIM n(f-1,2)
6100 '=========================================== LIENS ENTRE DEUX PLANS
6110 '=========================================================== Amorce
6120 b=-1
6130 b=b+1
6140 IF b=q(p+1) THEN PRINT"Erreur dans le plan "p:END
6150 GOSUB 6500
6160 FOR k=1 TO q(p)
6170 test1=n0*(p(p,k,0)-p(p,0,0))+n1*(p(p,k,1)-p(p,0,1))
6180 IF test1>0 GOTO 6130
6190 NEXT
6200 a0=p(p,0,0):a1=p(p,0,1):a2=z(p):fga=0
6210 b0=p(p+1,b+1,0):b1=p(p+1,b+1,1):b2=z(p+1):fgb=b+1
6220 c0=p(p+1,b,0):c1=p(p+1,b,1):c2=z(p+1):fgc=b
6230 t0=t
6240 n2=(a0-b0)*(c1-b1)-(a1-b1)*(c0-b0)
6250 IF n0<>0 THEN n2=n2*SGN(x*n0) ELSE n2=n2*SGN(y*n1)
6260 IF fgb=q(p+1) THEN fgb=-1
6270 fgb0=fgb
6280 CLS:PEN 1:GOTO 7400
6500 '==================== calcul du vecteur normal
6510 x=(z(p+1)-z(p))*(p(p+1,b,1)-p(p+1,b+1,1))
6520 y=(p(p+1,b+1,0)-p(p+1,b,0))*(z(p+1)-z(p))
6530 l=1
6540 test2=x*(p(p+1,l+b+1,0)-p(p+1,b+1,0))+y*(p(p+1,l+b+1,1)-p(p+1,b+1,1))
6550 IF test2=0 THEN l=l+1:GOTO 6540
6560 IF test2>0 THEN n0=-x:n1=-y ELSE n0=x:n1=y
6570 RETURN
```

```
7000 '========================================================== Algorithme
7010 NN0=n0:NN1=n1:NN2=n2
7020 x=p(p1,fga+1,0):y=p(p1,fga+1,1):z=z(p1)
7030 GOSUB 7200
7040 ps1=ps:n00=n0:n11=n1:n22=n2
7050 x=p(p2,fgb+1,0):y=p(p2,fgb+1,1):z=z(p2)
7060 GOSUB 7200
7070 ps2=ps
7080 IF ps1<ps2 GOTO 7120 ELSE n0=n00:n1=n11:n2=n22
7090 c0=a0:c1=a1:c2=a2:fgc=fga
7100 a0=b0:a1=b1:a2=b2:fga=fgb
7110 b0=p(p1,fgc+1,0):b1=p(p1,fgc+1,1):b2=c2:fgb=fgc+1:GOTO 7400
7120 c0=b0:c1=b1:c2=b2:fgc=fgb
7130 b0=p(p2,fgb+1,0):b1=p(p2,fgb+1,1):fgb=fgb+1:GOTO 7400
7200 '================== calcul de produit scalaire
7210 n0=(b1-a1)*(z-a2)-(b2-a2)*(y-a1)
7220 n1=(b2-a2)*(x-a0)-(b0-a0)*(z-a2)
7230 n2=(b0-a0)*(y-a1)-(b1-a1)*(x-a0)
7240 test3=n0*(c0-a0)+n1*(c1-a1)+n2*(c2-a2)
7250 IF test3>0 THEN n0=-n0:n1=-n1:n2=-n2
7260 IF test3<>0 GOTO 7320
7270 c=fgc
7280 c=c-1:IF c=-1 THEN c=q(p2)
7290 test4=n0*(p(p2,c,0)-a0)+n1*(p(p2,c,1)-a1)+n2*(c2-a2)
7300 IF test4=0 GOTO 7280
7310 IF test4>0 THEN n0=-n0:n1=-n1:n2=-n2
7320 r=n0*n0+n1*n1+n2*n2:r=SQR(r)
7330 ps=n0*NN0+n1*NN1+n2*NN2
7340 IF r<>0 THEN ps=ps/r
7350 RETURN
7360 GOTO 7000
7400 '===================================== iteration
7410 IF a2=z(p) THEN p1=p:p2=p+1 ELSE p1=p+1:p2=p
7420 IF fga=q(p1) THEN fga=-1
7430 IF fgb=q(p2) THEN fgb=-1
7440 IF fga=0 AND fgb=fgb0 AND p1=p AND t>1+t0 GOTO 7500
7450 GOSUB 9000
7460 GOTO 7000
7500 '================================================== Changement de plans
7510 p=p+1
7520 IF p=n GOTO 8000
7530 GOTO 6100
8000 '================================================== FIN.FACES EXTREMES
8010 f=t
8020 q0=q(0)
8030 q1=q(n)
8040 q=MAX(q1,q0):DIM fe(1,q,2)
8050 FOR u=0 TO q0
8060 FOR v=0 TO 1
8070 fe(0,u,v)=p(0,u,v)
8080 NEXT:fe(0,u,2)=z(0):NEXT
8090 FOR u=0 TO q1
8100 FOR v=0 TO 1
8110 fe(1,u,v)=p(n,u,v)
8120 NEXT:fe(1,u,2)=z(n):NEXT
8130 flag=1
8140 ERASE p:ERASE q
8150 GOTO 1300
9000 '================================== Enregistrement face+vecteur normal
9010 a(t,0,0)=a0:a(t,0,1)=a1:a(t,0,2)=a2
9020 a(t,1,0)=b0:a(t,1,1)=b1:a(t,1,2)=b2
9030 a(t,2,0)=c0:a(t,2,1)=c1:a(t,2,2)=c2
9040 n(t,0)=n0:n(t,1)=n1:n(t,2)=n2
9050 t=t+1
9060 RETURN
10000 '======================================================= Plot baladeur
10010 PLOT 0,0,y
10020 t=0
10030 WHILE INKEY(79)=-1 AND INKEY(58)=-1 AND INKEY(71)=-1 AND INKEY(63)=-1
 AND INKEY(19)=-1 AND INKEY(22)=-1 AND INKEY(68)=-1 :WEND
10040 IF INKEY(68)<>-1 THEN q(p)=t-1:RETURN
10050 IF INKEY(71)=0 THEN PLOT XPOS-2,YPOS,y
10060 IF INKEY(71)=32 THEN PLOT XPOS-4,YPOS,y
10070 IF INKEY(71)=128 THEN PLOT XPOS-1,YPOS,y
10080 IF INKEY(71)=160 THEN PLOT XPOS,YPOS,0:PLOT XPOS-2,YPOS,y:GOTO 10210
```

```
10090 IF INKEY(63)=0 THEN PLOT XPOS+2,YPOS,y
10100 IF INKEY(63)=32 THEN PLOT XPOS+4,YPOS,y
10110 IF INKEY(63)=128 THEN PLOT XPOS+1,YPOS,y
10120 IF INKEY(63)=160 THEN PLOT XPOS,YPOS,0:PLOT XPOS+2,YPOS,y:GOTO 10210
10130 IF INKEY(22)=160 THEN PLOT XPOS,YPOS,0:PLOT XPOS,YPOS-2,y:GOTO 10210
10140 IF INKEY(22)<>-1 THEN PLOT XPOS,YPOS-2,y
10150 IF INKEY(19)=160 THEN PLOT XPOS,YPOS,0:PLOT XPOS,YPOS+2,y:GOTO 10210
10160 IF INKEY(19)<>-1 THEN PLOT XPOS,YPOS+2,y
10170 IF INKEY(79)=0 THEN PLOT XPOS,YPOS,0:t=t-1:PLOT P(t,0,P),P(t,1,P),y:G
OTO 10210
10180 IF INKEY(58)<>-1 AND t=0 THEN PLOT XPOS,YPOS,2:t=1:P(P,0,0)=XPOS:
P(P,0,1)=YPOS:PRINT CHR$(7):GOTO 10210
10190 IF INKEY(58)<>-1 THEN xP=XPOS:yP=YPOS:P(P,t,0)=XP:P(P,t,1)=YP:
PLOT P(P,t-1,0),P(P,t-1,1),2:DRAW xP,yP:t=t+1:PRINT CHR$(7):GOTO 10210
10200 IF t>20 THEN PRINT"Stop":RETURN
10210 LOCATE 1,1:PRINT USING "##";t:LOCATE 1,25:PRINT XPOS;:PRINT"/"YPOS;
10220 GOTO 10030
10230 RETURN
12000 '================================================== Restitution
12010 MOVE 0,0
12020 PLOT P(P,0,0),P(P,0,1),y
12030 FOR t=1 TO q(P)
12040 DRAW P(P,t,0),P(P,t,1)
12050 NEXT
12060 PRINT CHR$(7)
12070 RETURN
```

Il est temps à présent de passer à l'explication détaillée du Programme VI.4. Ce programme, comme les deux précédents, est destiné à être rajouté au Programme VI.1 (et à ceux qui précèdent le VI.1 dans son explication).

Ce programme se décompose en trois parties principales. La première (lignes inférieures à 1000) est composée de lignes qui s'insèrent dans le Programme VI.1 pour assurer sa bonne exécution avec des polyèdres étagés. Nous y reviendrons lorsque nous aurons mieux compris la manière dont ils sont engendrés, celle-ci conditionnant évidemment leur utilisation.

La deuxième partie, nommée *plans étagés*, est destinée à engendrer les $n + 1$ intersections avec les plans T_i. Ces intersections sont données à l'aide du plot baladeur, qui a été quelque peu modifié pour ne donner que des polygones ; le premier point est toujours enregistré séparé, les autres joints au précédent. Les pressions successives de la touche "*e*", après déplacement de plot, engendreront donc une ligne brisée. D'autre part, le sous-programme *restitution* redessine, avant d'enregistrer une intersection, toutes les intersections précédentes ; j'aurai l'occasion de dire, au paragraphe suivant, l'utilité de ce système.

Précisons un peu les notations. $P_{u,v}$ est le v^{ieme} point de la u^{ieme} intersection (avec v ou u nul éventuellement, comme toujours), et $p(u, v, w)$ est sa

w^{ieme} coordonnée. La matrice p étant dimensionnée à $(n, 20, 1)$, on ne peut mettre que 21 points par intersection, pour cause de place mémoire. D'autre part, la troisième coordonnée du point n'est pas $p(u, v, 2)$, mais $z(u)$, puisque T_u a pour équation $z = z(u)$.

Sur les intersections elles-mêmes, le programme se charge de les fermer, c'est-à-dire qu'il rejoint le dernier point avec le premier. Il est précisé qu'elles doivent être convexes, sinon le PCX ne saurait admettre un polygone concave pour face, et le programme aboutirait à une erreur. En outre, pour des raisons techniques, le polygone doit tourner dans le sens trigonométrique, et l'angle polaire du point $P_{p+1,0}$ doit être inférieur à celui de $P_{p,0}$; nous allons voir pourquoi.

Passons à la partie principale du programme, la *génération du polyèdre.*

Ce programme, comme le VI.3, a un gros avantage en tout cas : c'est que la génération du polyèdre exige de calculer le vecteur normal de chaque face. Comme nous le verrons au paragraphe suivant, ces vecteurs sont extrêmement utiles, et même nécessaires à la gestion des parties cachées. Or, ils seront ici déjà tout calculés. C'est pourquoi on conserve leurs trois coordonnées dans la matrice n.

Passons à la partie *liens entre deux plans*. Il s'agit d'engendrer le PCX_p, c'est-à-dire ce polyèdre convexe imaginaire qui joint les deux plans T_p et T_{p+1}. Cette opération sera répétée pour tous les interplans, d'où la partie *changement de plans* (lignes 7500 et suivantes).

Nous avons ici deux polygones, et nous cherchons les faces qui les joignent (voir Figure VI.E,*c*). Pour cela, nous utilisons un algorithme, précédé d'une amorce.

L'algorithme consistant à trouver une face à partir de la précédente, l'initialisation doit nécessairement trouver une première face. Pour cela, on cherche l'une des faces contenant le point $P_{p,0}$. Dans la suite, nous supposerons que tous les polygones sont de *vrais* polygones, c'est-à-dire ont au minimum trois points(1). Dès lors, il existe au moins un nombre b tel que $P_{p,0} P_{p+1,b} P_{p+1,b+1}$ soit une face. Il ne reste plus qu'à le trouver, en essayant tous les b possibles (sauf $b = q(p+1)$ et $b + 1$ "=" 0 pour des raisons techniques ; à votre avis, lesquelles et comment les résoudre ?) *(Question VI.d.)* La face est bonne, en principe, si tous les autres sommets sont du même côté. En réalité, il suffit de s'assurer que tous les points de

(1) A votre avis pourquoi ? Comment faire autrement ? *(Question VI.c).*

T_p sont du même côté que ceux de T_{p+1}, car le polygone Pg_{p+1}, étant convexe et ayant une arête commune avec la face à tester, est forcément tout entier du même côté. C'est ici que la convexité des polygones devient essentielle ; sans elle, il risque de ne pas y avoir de *b*, donc pas d'amorce, donc... rien.

Ce *b* étant trouvé, nous avons enfin notre première face. Il faut alors se préparer à l'algorithme.

Celui-ci fonctionne à partir d'une face ABC organisée comme suit. Les trois points ABC joignant les deux polygones Pg_p et Pg_{p+1}, il est obligatoire que l'un des points soit d'un côté, et les deux autres de l'autre. Ces derniers, de plus, formant une arête, sont forcément successifs. L'algorithme appelle A le point isolé, p_1 le numéro de son polygone ($p_1 = p$ ou $p_1 = p + 1$), et fg_a (flag *a*) son numéro d'ordre en son sein. Donc $A = P_{p_1, fg_a}$. Quant aux points B et C, ils se suivent sur le polygone p_2, de sorte que $fg_b = fg_c + 1$. On voit la nécessité que les polygones soient orientés dans le même sens, car l'algorithme progresse justement dans ce sens-là, quelle que soit la valeur de p_1. Dès lors, AC est nécessairement l'arête extrême de la face enregistrée juste avant ABC.

Il en résulte (et c'est encore plus clair sur la Figure VI.E,*c*) que la face suivante, à rechercher, admet AB pour arête. On voit que, contrairement au PCM, le choix de l'arête de *recherche* n'est pas ici arbitraire, au contraire.

Mais il y a mieux : le point manquant D, tel que ABD soit une face, n'est pas du tout arbitraire non plus. Alors que, pour le PCM, ce pouvait être pratiquement n'importe quel point, ici on n'a le choix qu'entre deux.

Il s'agit en effet, obligatoirement, soit du "suivant de A", Q, soit du "suivant de B", P. Il n'y a pas d'autre possibilité, ainsi que vous vous en rendrez compte en méditant sur la figure.

Dès lors, le reste est assez clair. On exécute le test des produits scalaires qui a été décrit pour le PCM. On sait ensuite par comparaison quel point est le bon, il ne reste qu'à enregistrer la face et à se préparer à engendrer la suivante. Pour cela, il faut redéfinir A, B, C. Ainsi, si Q est le *bon* point, alors c'est B qui est isolé dans la nouvelle face BQA. Donc B devient A, A devient C, car il précède Q qui devient B. Si c'est P le bon point, alors A reste isolé, B devient C et P devient B. Notez que, dans les deux cas, C passe aux oubliettes...

L'algorithme s'arrête lorsque la face obtenue est égale à la face de départ. Pour en être sûr, on conserve en mémoire la valeur de fg_b de

l'amorce (fg_{b_0}). Sur l'amorce, en outre, $fg_a = 0$ et $p_1 = p$. Quand ces trois valeurs se retrouvent identiquement répétées, on est revenu au point de départ, il n'y a donc plus qu'à changer d'interplan.

Enfin, lorsque p + 1 dépasse *n*, on ne peut continuer ; la génération du polyèdre est terminée.

Toutefois, dans le polyèdre final, le premier et le dernier polygones sont aussi des faces, que nous appellerons faces extrêmes. Ces faces, très simples à engendrer puisqu'il n'y a aucune opération à exécuter, posent toutefois un léger problème. Comme il faudra bien qu'elles soient conservées, il faut soit garder la matrice *p* (mais cela prend inutilement de la place car elle ne sert plus à rien), soit les insérer dans la matrice *a*, qui est prévue pour des faces triangulaires ; une *élongation* de cette matrice à *q* points (définition de *q* aux lignes 8020-8040) utiliserait aussi beaucoup de place en mémoire. Ces deux solutions étant mauvaises, j'en ai choisi une troisième, consistant à créer une matrice spéciale *fe* pour contenir ces deux faces abandonnées. Évidemment, cela exige de modifier un peu les programmes de dessin du polyèdre, etc., car il ne faut pas laisser ces orphelines à la traîne, d'où la première partie du programme (lignes inférieures à 1000), causée par la matrice *fe*. Mais la présence de cette matrice permet d'éliminer *p* et *q*, devenues inutiles (ligne 8140). Une économie de mémoire est toujours bonne à prendre.

En tout cas, ça marche. Essayez donc !

8. SUPPRESSION DES PARTIES CACHEES DES POLYEDRES CONVEXES

A force de m'entendre parler de la gestion des parties cachées, le lecteur doit être bien impatient de voir de quoi il retourne !

Tout d'abord, pour être convaincu de son utilité, il suffit, je pense, de regarder les Figures VI.F et VI.F *bis*. Sur la première, vous voyez essentiellement ce que l'on peut appeler un beau fouillis. Ce n'est pourtant rien d'autre que le gros polyèdre de l'autre figure, qui est représenté d'abord toutes faces visibles, ensuite “avec parties cachées”, c'est-à-dire en fait au naturel. Il va sans dire que la seconde est plus agréable à l'œil, et plus réaliste que l'autre.

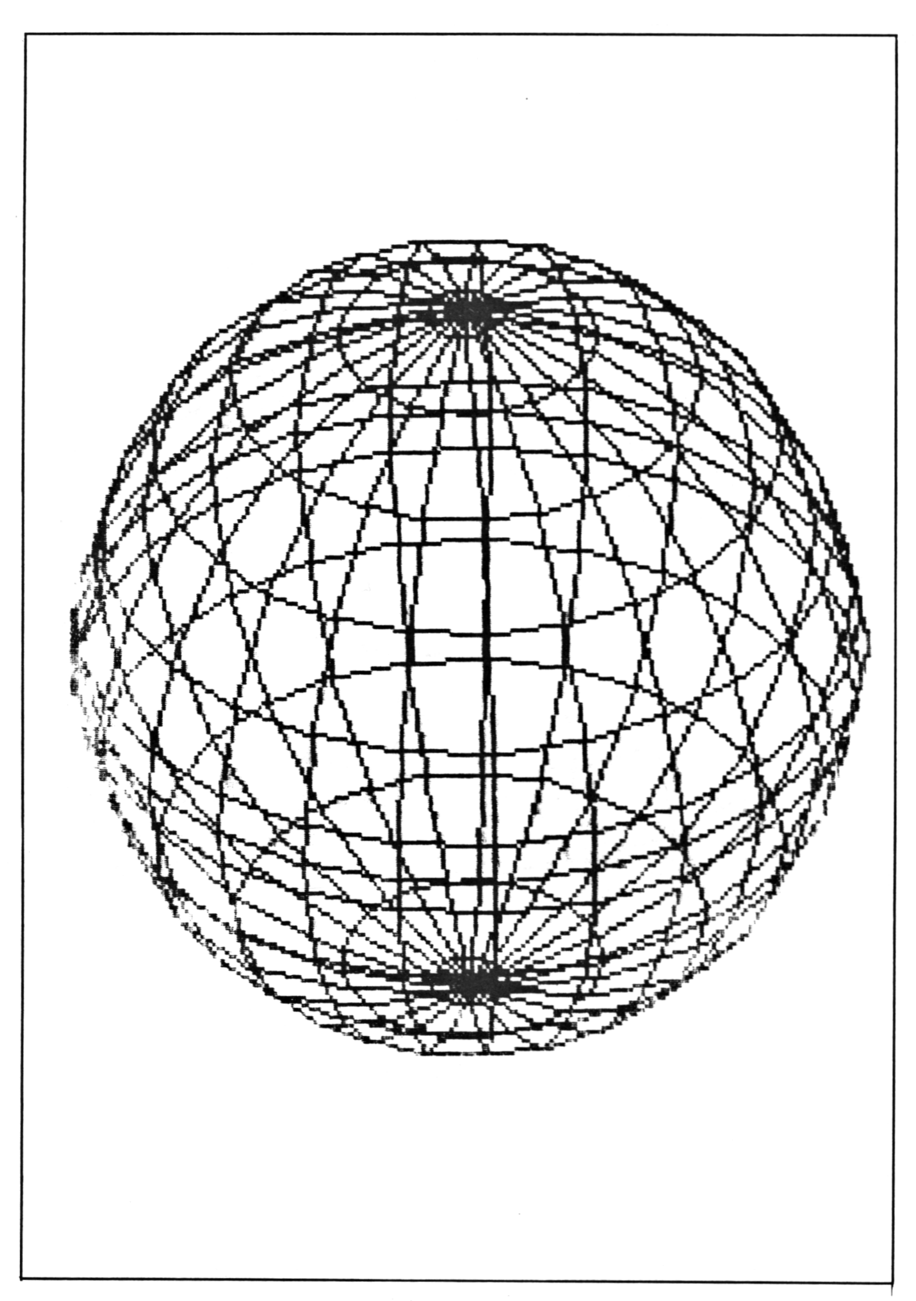


Figure VI.F

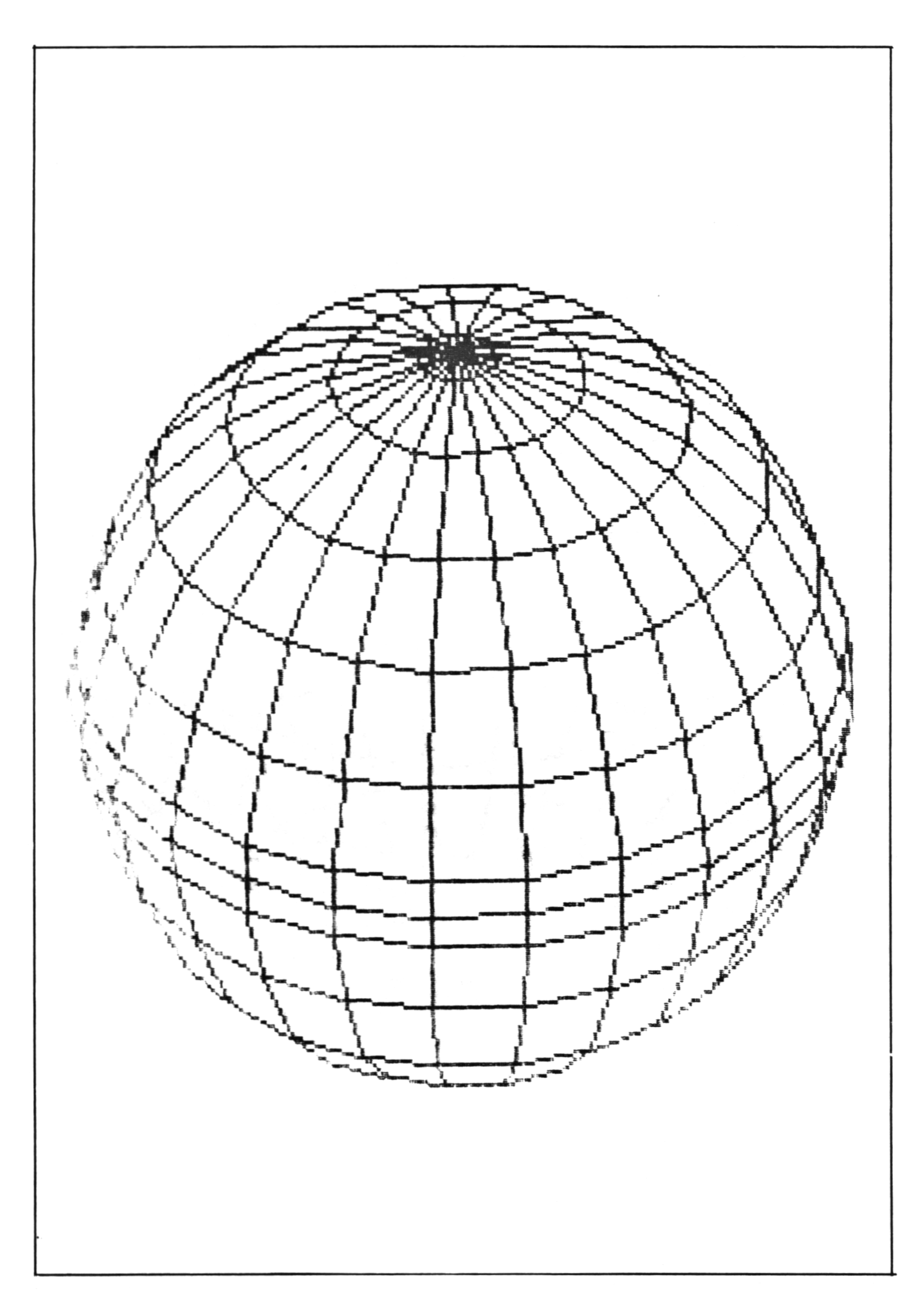


Figure VI.F bis

En effet, lorsque vous regardez un polyèdre, par exemple un dé, ou plus généralement n'importe quel objet, vous ne pouvez, sans miroir, le voir intégralement. Ainsi, pour un dé, autrement dit un cube, vous ne pouvez voir plus de trois faces à la fois, sauf s'il est en verre (et encore). Cela, c'est la réalité. Mais ce n'est pas ce que vous avez vu jusqu'à présent sur votre Amstrad, qui vous a bêtement et indistinctement tracé toutes les faces de chacun des polyèdres que vous avez définis (ce qui, en fait, n'est déjà pas mal).

Nous allons tenter à présent de remédier à ce manque, afin d'obtenir des polyèdres, et plus tard des objets plus complexes, qui soient quelque peu réalistes.

J'ai déjà eu l'occasion d'expliquer à quel point cela pouvait être complexe pour les polyèdres concaves, et pour quelles raisons.

Pour le moment, c'est donc simplement des polyèdres convexes que nous allons nous occuper. Cela ne sera d'ailleurs pas très long car la méthode est extrêmement simple. Observez la Figure VI.E, schéma *d*. On y voit un polyèdre convexe, l'écran E sur lequel on le projette, avec votre œil derrière. Cet écran a un vecteur normal que l'on oriente *sortant*, c'est-à-dire la pointe vers l'œil, que nous appellerons **V**. Je précise tout de suite que la norme **V** est indifférente. Seuls comptent sa direction et son sens.

D'autre part, chaque face a un vecteur normal (double flèche). A côté de chacun de ses vecteurs normaux (dont la norme est également indifférente), nous voyons un représentant du vecteur **V**, ce qui permet de mettre en valeur l'angle qui sépare ces deux vecteurs. Or, constatant que, pour l'œil, seules les deux faces de droite sont visibles, nous pouvons faire l'observation suivante sur ces angles.

Soit l'angle est obtus (face marquée d'un 1), et la face est invisible. Soit l'angle est aigu (face 2), et la face est visible. Soit (cas limite de la face 3) l'angle est droit, et la face est les deux à la fois : elle est vue par la tranche. Comme cela ne rajoute rien au dessin à faire sur l'écran, nous pouvons la considérer comme invisible.

Nous savons à présent sur quoi baser notre appréciation. Mais plutôt que de calculer cet angle, ce qui serait pénible, nous pouvons ajouter cette remarque : si l'on appelle prod le produit scalaire de **V** et du vecteur normal de la face considérée, ce prod est du même signe que le cosinus de l'angle séparant les deux vecteurs (voir Chapitre II). Il est donc positif si l'angle est aigu (face visible), négatif sinon. Il suffit donc de calculer prod

et de voir son signe. Or, quoi de plus facile à calculer qu'un produit scalaire ?

Le seul problème qui subsiste, si l'on peut dire, c'est la présence des vecteurs normaux, qu'il faut calculer. C'est pourquoi j'ai tenu à conserver ceux qui sont automatiquement créés par le Programme VI.4. C'est là la supériorité de ce programme sur le Programme VI.2, où les vecteurs normaux ne sont pas calculés.

PROGRAMME VI.5

```
300 '========================================== Calcul des vecteurs normaux
310 DIM n(f-1,2)
320 FOR t=0 TO f-1
330 a0=a(t,0,0):a1=a(t,0,1):a2=a(t,0,2)
340 b0=a(t,1,0):b1=a(t,1,1):b2=a(t,1,2)
350 c0=a(t,2,0):c1=a(t,2,1):c2=a(t,2,2)
360 x=(b1-a1)*(c2-a2)-(b2-a2)*(c1-a1)
370 y=(b2-a2)*(c0-a0)-(b0-a0)*(c2-a2)
380 z=(b0-a0)*(c1-a1)-(b1-a1)*(c0-a0)
390 IF t=f-1 THEN te=0 ELSE te=t+1
400 xte=a(te,0,0):yte=a(te,0,1):zte=a(te,0,2)
410 Testeur=(xte-a0)*x+(yte-a1)*y+(zte-a2)*z
420 IF Testeur >0 THEN x=-x:y=-y:z=-z
430 n(t,0)=x
440 n(t,1)=y
450 n(t,2)=z
460 NEXT
625 Prod=c1*s2*n(t,0)+s1*s2*n(t,1)+c2*n(t,2)
630 IF Prod<=0 GOTO 720
```

PROGRAMME VI.6

```
10 '========================================================= Tous Polyedres
20 '==================================================== Entree des donnees
30 MODE 1:PRINT"Le Polyedre est-il a symetrie""de revolution?"
35 DEG
40 WHILE INKEY$="":WEND
50 IF INKEY(34)<>-1 GOTO 3000
60 CLS:PRINT"Voulez-vous definir le Polyedre Par"
70 PRINT"des Plans etages?"
80 WHILE INKEY$="":WEND
90 IF INKEY(34)<>-1 GOTO 5000
100 CLS:PRINT"Le Polyedre est donc quelconque."
110 FOR t=0 TO 3000:NEXT
120 CLS
485 IF flag=1 THEN umax=2 ELSE umax=AA(t)-1
490 FOR u=0 TO umax
521 IF flag=0 GOTO 530
632 IF flag=1 THEN umax=2 ELSE umax=AA(t)-1
635 FOR u=0 TO umax
725 IF flag=0 GOTO 2000
3665 flag=0
```

N'importe, le Programme VI.5 suppléera à ces insuffisances. Les lignes 300 à 460 permettent de calculer les vecteurs normaux quand ce n'est déjà fait (d'où la ligne 305 du Programme VI.4). Peu de commentaires sur ce point, car nous l'avons déjà vu en Figure VI.E,*b*. Notez seulement que le Programme VI.2 est conçu de telle sorte que le premier point d'une face n'appartient pas à la précédente, et peut donc servir de point D de test.

Quant aux lignes 625 et 630, ce sont elles qui assurent la gestion des parties cachées, en calculant prod et en déterminant son signe. Si celui-ci est négatif, alors la face ne doit pas être tracée, on passe à la suivante.

A présent, il ne vous reste plus qu'à amalgamer le Programme VI.1, précédé comme il convient, avec les Programmes VI.2, VI.4, VI.5 et VI.6 (dans cet ordre) pour obtenir un très gros programme : *tous polyèdres*, qui cumulera à peu près tout ce que nous avons vu jusqu'à présent. Il sera parfaitement complet si vous souhaitez, après la ligne 120, rajouter les INPUT permettant de rentrer un polyèdre quelconque ne pouvant l'être avec les deux méthodes jusqu'ici exposées. Si vous avez eu le courage de réaliser le Programme VI.3, rien ne vous empêche de l'insérer en supplément !

Avant de vous laisser à de polyédriques chimères, je dois encore penser au lecteur qui se demande avec angoisse comment il pourra savoir si son polyèdre est convexe ou non, à partir des systèmes “Polyèdres à répétition cylindrique” et “Polyèdres étagés”.

Nous avons vu que, pour ces derniers, il était nécessaire que les polygones soient convexes. Un polygone convexe n'est pas difficile à reconnaître, avec un peu d'habitude. Au début, dessinez votre polygone en couleur (en rouge, par exemple). Puis, en noir, joignez chaque sommet à tous les autres, sans repasser les traits rouges. Si tous les traits noirs sont à l'intérieur de la ligne rouge, votre polygone est convexe, c'est tout simple.

Mais, pour les polyèdres étagés, ce n'est pas suffisant. Le polyèdre obtenu ne sera convexe qu'à d'autres conditions. En particulier, deux polygones successifs doivent être contenus l'un dans l'autre (c'est pour cela que le Programme VI.4 redessine les polygones précédents). De plus, cela doit croître, puis décroître, c'est-à-dire que le deuxième doit contenir le premier, le troisième le deuxième, etc., puis le sens s'inverser (mais *une* seule fois) de telle sorte qu'à la fin c'est le dernier qui est inclus dans l'avant-dernier. Ensuite votre polyèdre sera peut-être convexe, surtout si vous avez pris la précaution d'espacer beaucoup (grosse différence d'ordonnées) deux polygones successifs dont l'un est nettement plus gros

que l'autre. Je sortirais de mon sujet en en disant davantage, mais je pense que cela devrait suffire.

Pour les polyèdres à répétition cylindrique, c'est plus simple : il faut et il suffit, pour assurer leur convexité, que la *baleine* soit convexe par rapport à l'axe O*z*, ce qui se vérifie ainsi : considérez l'extrémité supérieure A de la baleine, et *a* sa projection orthogonale sur O*z* puis de même B l'extrémité inférieure et *b*. Dès lors, l'ensemble (segment *ab* (sur O*z*) + segment *b*B + *baleine* + segment A*a*) forme une ligne brisée fermée.

On dit que la baleine est *convexe par rapport à Oz*, si cette ligne brisée fermée est un polygone convexe. Notez que cela pourrait même ne pas être un polygone, si la baleine traverse l'axe O*z*.

Vous voilà à présent paré pour tout ce qui touche les polyèdres, ou presque. Mais c'est seulement plus tard que nous aurons l'occasion de parler des parties cachées des polyèdres concaves, qui sont assez difficiles. Pour le moment, nous allons étudier un sujet complémentaire des polyèdres, les surfaces bombées.

REPONSES AUX QUESTIONS POSEES DANS CE CHAPITRE

Question VI.a

Tout simplement en modifiant le programme de tracé, pour qu'il ne trace que les deux premières arêtes de chacune des faces tétragonales, et même seulement la première pour les faces tétragonales du bas. En effet, sur les faces tétragonales, la deuxième arête est commune à la face située en dessous, la troisième à celle qui suit, et la dernière à celle du dessus.

Question VI.b

Il suffit, chaque fois que l'on trouve une nouvelle face ABC, de calculer les produits scalaires de AB avec tous les vecteurs normaux des faces précédentes. Lorsqu'un tel produit se trouve nul, la face dont on a pris le vecteur normal est parallèle à ABC. Reste à vérifier qu'elles sont bien sur le même plan (car elles peuvent être seulement parallèles), puis à les amalgamer.

Question VI.c

Pour tester la validité d'une face, le programme a besoin d'un point sur l'un des polygones qui n'appartienne pas à la face à tester. C'est obligatoire si les polygones ont au moins trois sommets, mais si l'un d'entre eux n'en a que deux, ou un seul, cela risque de poser de graves problèmes. Deux solutions sont possibles :

- Changer le programme avec des routines spéciales, mais c'est un peu lourd.
- Considérer un troisième point, très proche du premier (différence de 0,01), donc invisible, mais qui aidera le programme (c'est le même principe que pour les faces triangulaires des polyèdres à répétition cylindrique).

Question VI.d

Le point suivant de b est $b + 1$, sauf si $b = q(p)$. Dans ce cas, il faudrait remplacer dans tout le programme les $b + 1$ par des b_1, et taper :

```
IF b = q (p) THEN b1 = 0 ELSE b1 = b + 1
```

en ligne 6140.

VII

TOUJOURS PLUS LOIN, TOUJOURS MIEUX

Après avoir vu des méthodes très rigoureuses et superbes, et obtenu finalement un échec partiel puisque nous ne savons pas encore tracer des polyèdres concaves sans parties cachées, nous allons faire ici l'éloge de trucs simplificateurs dus à certaines particularités des objets à tracer. Cet éloge sera soutenu par deux exemples un peu difficiles, mais très intéressants.

Le premier permet de tracer en trois dimensions un appartement, un labyrinthe, ou d'une façon générale toute pièce ou ensemble de pièces aux murs verticaux. Au passage, nous expliquerons comment, par extension, il est possible de gérer les parties cachées des polyèdres concaves, mais il n'y aura pas de programme (il risquerait de prendre des dimensions astronomiques !). Le programme pour l'"appartement" est déjà bien assez long, mais vous verrez qu'il est assez puissant ; or ce genre de chose peut toujours servir.

Le second exemple traite des surfaces "bombées", c'est-à-dire non planes. L'usage de ces surfaces apporte un complément indispensable aux polyèdres. Leur combinaison, comme il sera expliqué, est le fondement de la fameuse "C.A.O" (Conception Assistée par Ordinateur), qui permet de tracer ainsi à peu près n'importe quoi, sans parties cachées. C'est pourquoi j'ai jugé utile de traiter de cette question, quoique le sujet soit plutôt aride. Aussi ne vous inquiétez pas si tout n'est pas très clair, et n'hésitez pas à relire plusieurs fois, voire à lire les chapitres suivants (moins difficiles) avant de revenir à celui-ci. Je crois néanmoins que le jeu en vaut la chandelle, et qu'il eût été dommage de ne pas parler de ces surfaces.

1. S'ADAPTER AUX PARTICULARITES DES OBJETS TRACES

Il est rare qu'il existe une méthode générale absolue pour toute une catégorie d'objets. Même lorsqu'elle existe, elle est souvent si longue qu'il vaut mieux, effectivement, s'adapter aux particularités des objets tracés. Nous avons déjà vu cela avec les polyèdres à répétition cylindrique et les polyèdres étagés. Plus l'objet à tracer est susceptible d'être complexe, et plus cette règle s'impose.

Nous allons en voir deux exemples, mais souvenez-vous que l'on peut presque toujours trouver une simplification. C'est encore plus le cas, naturellement, dans un jeu où il se trouve quelques décors seulement, ou quelques "vaisseaux spatiaux", etc.

Prenons un exemple concret. Imaginons un jeu avec des chars de combat, tous les mêmes. Un char peut, en gros, être considéré comme un polyèdre, convexe de surcroît, et même étagé à la rigueur, à condition qu'on en omette un détail : le canon. Ce canon, d'autre part, est un petit cylindre (par le rayon), assez long, et mobile. Pourquoi dès lors ne pas tracer le corps de la machine, d'une part, comme un polyèdre étagé, puis le canon, d'autre part, comme un cylindre (voir Paragraphe 3 pour le tracé) ? Pour parachever le travail, il suffit d'effacer les traits du polyèdre situés derrière le canon, et voilà un char très convenable, en utilisant deux des tracés rendus possibles dans ce livre.

Pour vous convaincre totalement, nous allons appliquer tout cela à des exemples précis dont j'ai parlé en introduction.

2. LE LABYRINTHE OU L'APPARTEMENT DE REVE

Le long Programme VII.1 va permettre de créer en trois dimensions des objets particuliers. Lesquels ? Ce sont des objets obtenus par élévation. Cela signifie que, à partir d'un plan au sol constitué de segments, le programme va engendrer un objet en trois dimensions, chaque segment devenant un *mur* dont la base est la même que le segment, et dont les côtés sont verticaux et de longueur donnée au préalable. L'objet ainsi obtenu pourra ensuite être déplacé comme pour les polyèdres. Cela pourra être un appartement, par exemple, si vous en entrez le plan, un labyrinthe, ou tout autre ensemble de murs ; on en comprend dès lors l'usage, forcément assez

répandu dans les jeux (en outre, l'appartement sera plus mobile que ceux, visibles dans certains jeux, où les murs sont résolument fixes et vus toujours selon le même angle), et de surcroît, cela peut même servir professionnellement. Voyez ainsi la Figure VII.A.

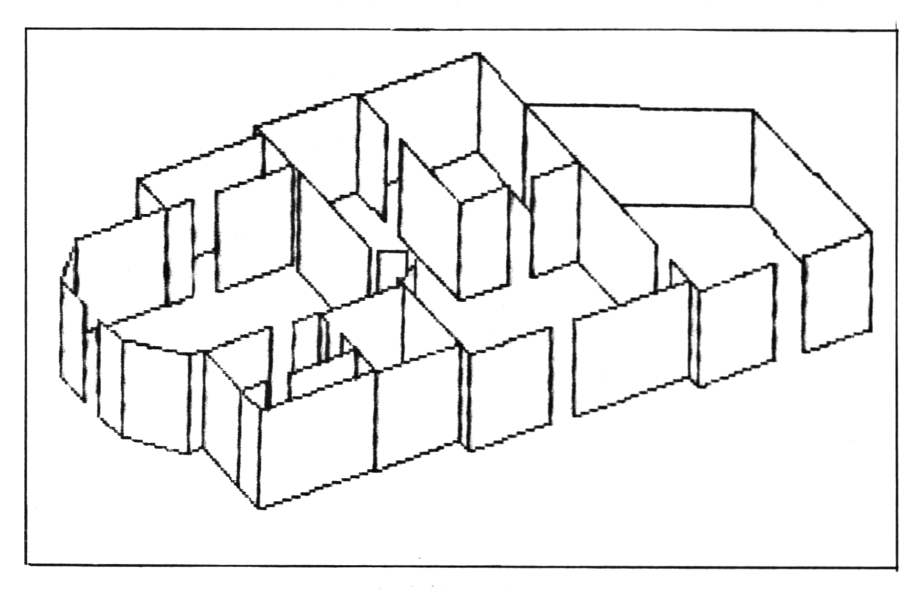


Figure VII.A

Cet objet, nous l'appellerons à présent l'appartement, sans nous soucier de son usage réel, puisque ce n'est pas le but que nous poursuivons. Vous voyez que je tiens tout de même à rester assez général.

Mais pas trop tout de même. Les particularités essentielles de ce type d'objets sont d'intérêt pratique et de commodité informatique. L'intérêt pratique réside dans leur conception : un plan au sol suffit. Nous savons comment le faire facilement, grâce au plot baladeur. Ainsi les premières lignes du Programme VII,1 sont-elles très largement inspirées, voire recopiées, du Programme IV.1.

Pour expliquer la commodité informatique, il faut pouvoir se repérer. Comparons donc avec les polyèdres concaves généraux. Par rapport à ceux-ci, nous semblons y perdre. Car, non seulement notre appartement hérite de leurs tares (car les murs peuvent également être partiellement cachés, et donc être tracés en mille morceaux), mais, en outre, il n'a même pas l'avantage d'être un polyèdre car il n'a ni plancher ni plafond, donc pas d'intérieur et d'extérieur, et certaines faces (murs en fait) sont au centre de

plusieurs autres. En réalité, quoique le principe soit globalement le même, l'appartement est nettement plus facile à tracer en trois dimensions. En voici les raisons :

1. Alors que les faces d'un polyèdre peuvent avoir quarante côtés, un mur n'en a que quatre, de toute façon.
2. De ces quatre côtés, deux sont verticaux, ce qui est infiniment précieux, comme nous le verrons dans un instant.
3. Les murs sont toujours des rectangles ; leur projection sera donc toujours un parallélogramme, ce qui est aussi précieux.

Pour mieux nous expliquer, nous allons distinguer trois parties dans le tracé de l'appartement. La première est constituée de toutes les arêtes qui feraient partie du plafond, s'il y en avait un : c'est la partie haute. La deuxième est formée de toutes les arêtes qui feraient partie du plancher, s'il y en avait un : c'est la partie basse. La partie intermédiaire est formée par toutes les arêtes verticales des murs.

Nous admettrons les propriétés suivantes :

- Toutes les arêtes de la partie haute sont entièrement visibles lorsque l'appartement est vu par en haut (ce que nous supposerons).
- Toutes les arêtes verticales sont en un seul morceau ; elles ne sont pas forcément entièrement visibles mais, de toute façon, si elles sont coupées, c'est forcément par une arête de la partie haute, et dès lors leur partie supérieure est visible, le reste ne l'est plus.
- Seules les arêtes de la partie inférieure posent un problème : elles peuvent être visibles en de nombreux morceaux, il n'y a pas de cas vraiment général. Mais cela ne fait qu'une arête sur quatre, alors que dans les polyèdres concaves toutes les arêtes étaient susceptibles d'être coupées en morceaux.
- Dernier avantage enfin : sauf lorsque l'appartement est vu par la tranche, aucun mur ne peut en dissimuler complètement un autre, alors que cela arrive sur les polyèdres, et c'est extrêmement ennuyeux car difficile à repérer.

Malgré tous ces avantages, le tracé de l'appartement est encore assez long (et évidemment d'autant plus qu'il y a plus de murs). C'est pourquoi, afin de maintenir malgré tout l'animation, j'ai pris le parti d'exécuter d'abord tous les calculs (en les stockant ensuite), puis seulement d'effacer l'écran et de tracer le résultat. Ne vous étonnez donc pas si, au cours de

l'exécution du programme, les touches semblent sans effet ; en fait, le calcul est en cours.

Mais de quel calcul s'agit-il ? Pour mieux me faire comprendre, je m'appuierai sur la Figure VII.B.

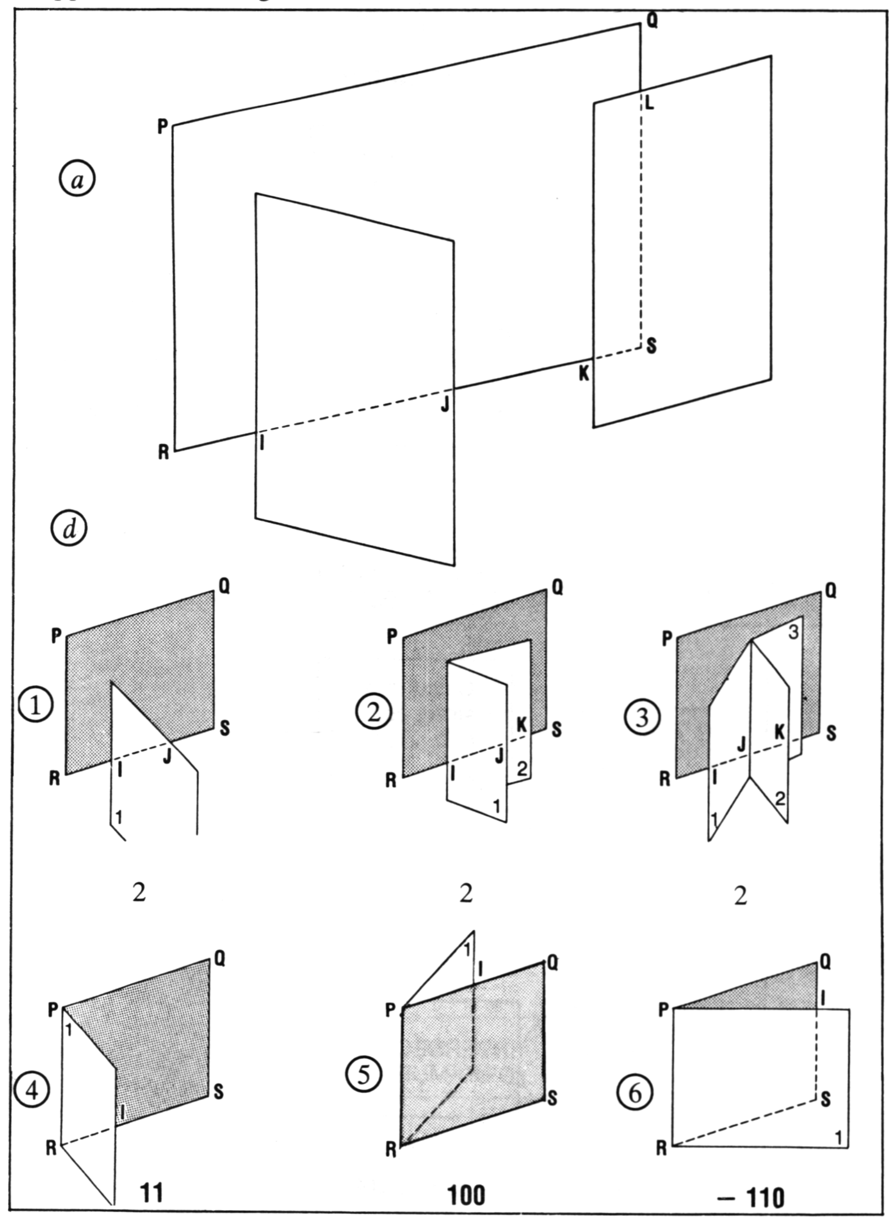
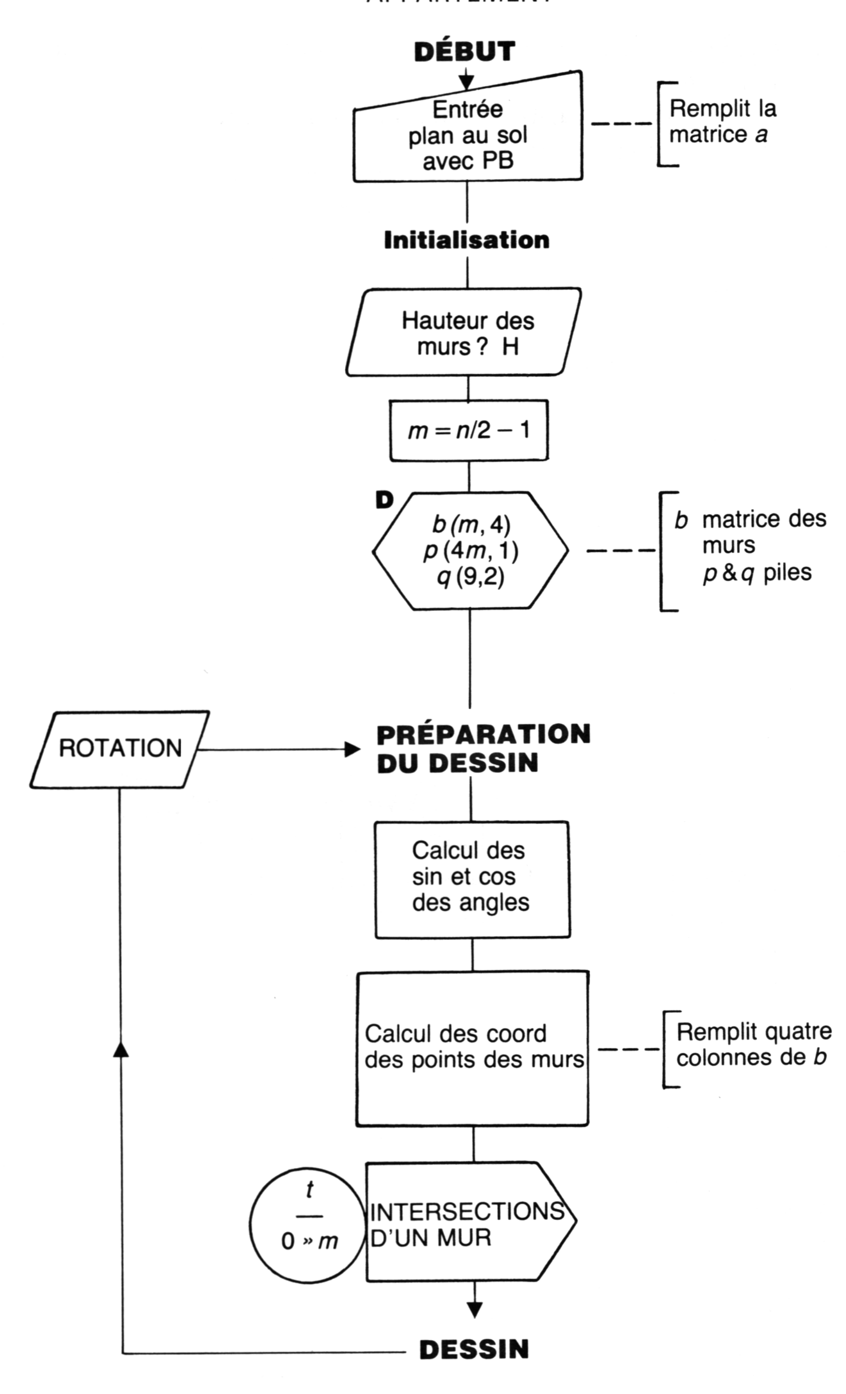
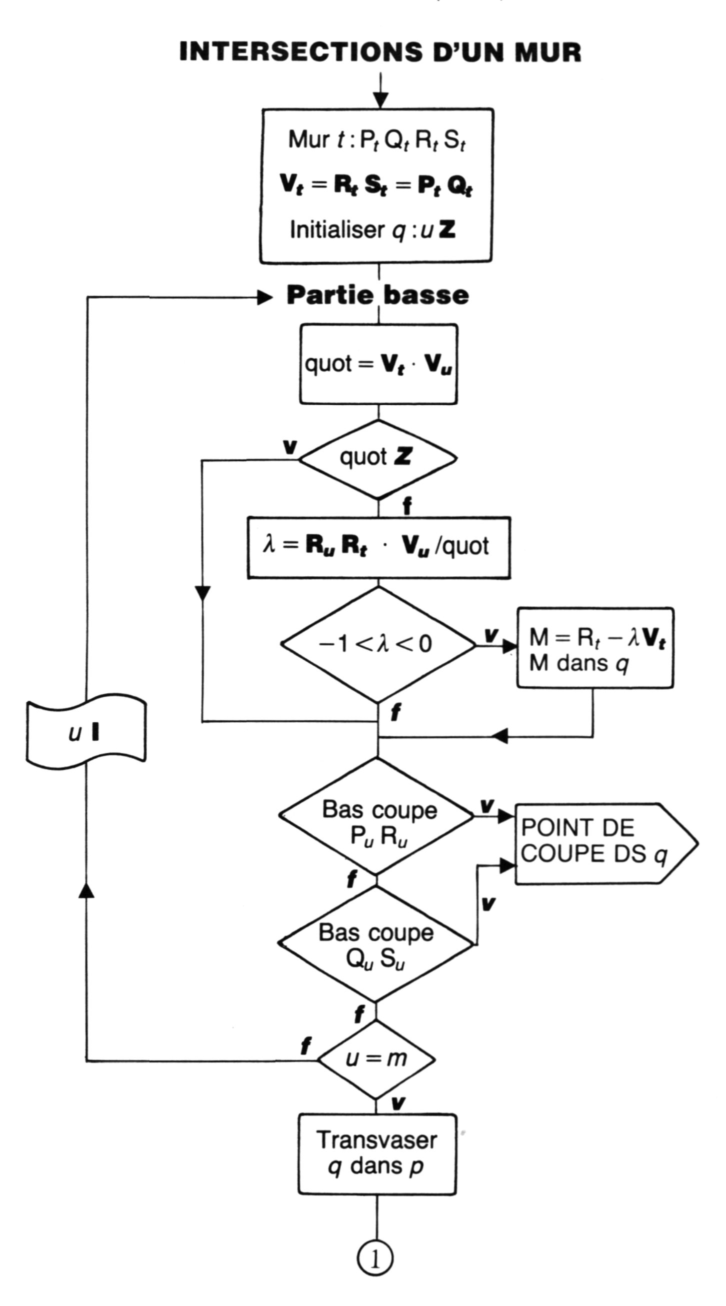
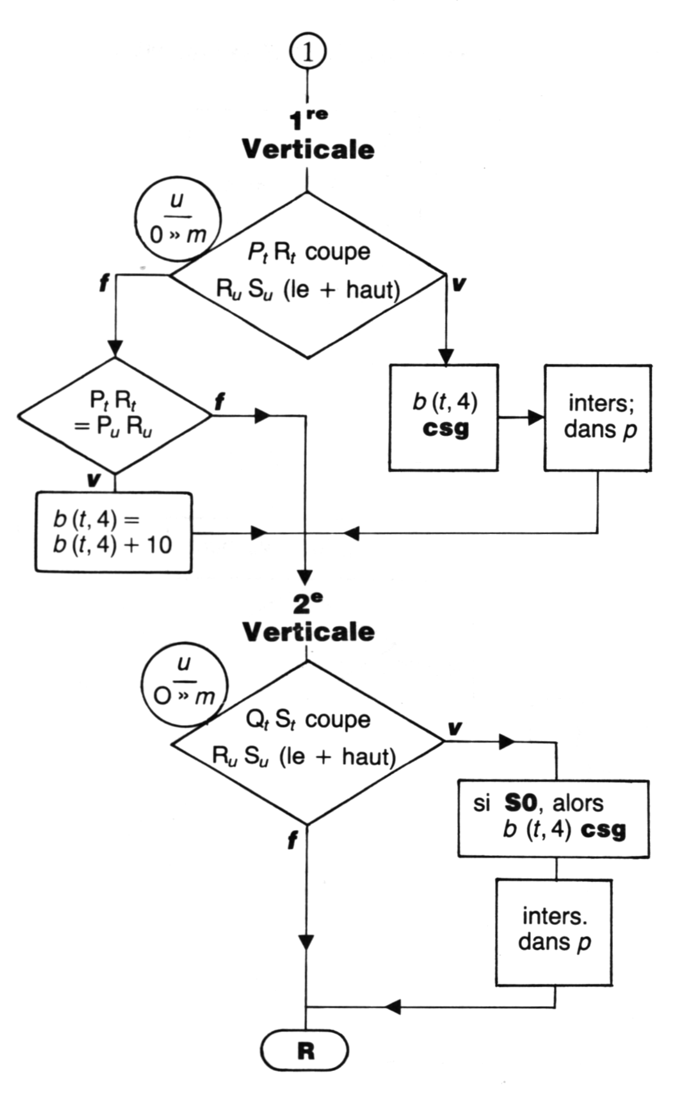


Figure VII.B

APPARTEMENT




INTERSECTIONS D'UN MUR
Mur t : Pt Qt Rt St
Vt = Rt St = Pt Qt
Initialiser q : u Z
Partie basse
quot = Vt · Vu
quot Z
v
f
λ = Ru Rt · Vu /quot
−1 < λ < 0
v
M = Rt − λVt
M dans q
f
u I
Bas coupe
Pu Ru
v
POINT DE
COUPE DS q
f
v
Bas coupe
Qu Su
f
f
u = m
v
Transvaser
q dans p
1


1
1re
Verticale
u
0 » m
Pt Rt coupe
Ru Su (le + haut)
f
v
Pt Rt
= Pu Ru
f
v
b (t, 4) =
b (t, 4) + 10
b (t, 4)
csg
inters;
dans p
2e
Verticale
u
O » m
Qt St coupe
Ru Su (le + haut)
v
f
si S0, alors
b (t, 4) csg
inters.
dans p
R

PROGRAMME VII.1

```
10 '============================================================ Plot baladeur
20 MODE 1:INK 0,1:BORDER 1:INK 1,24:PEN 1:INK 2,6
30 DIM a(200,1)
40 DEFINT h,i,u,v,t
50 ORIGIN 320,200
70 PLOT 0,0,1:t=0
80 WHILE  INKEY(79)=-1 AND INKEY(58)=-1 AND INKEY(74)=-1 AND INKEY(75)=-1
AND INKEY(72)=-1 AND INKEY(73)=-1 AND INKEY(68)=-1 AND INKEY(77)=-1:WEND
90 IF INKEY(68)<>-1 THEN n=t:GOTO 500
100 i=INKEY(74):IF i<>-1 THEN v=0:u=-2-i/16:GOSUB 300
110 i=INKEY(75):IF i<>-1 THEN v=0:u= 2+i/16:GOSUB 300
120 i=INKEY(73):IF i<>-1 THEN u=0:v=-2-i/16:GOSUB 300
130 i=INKEY(72):IF i<>-1 THEN u=0:v= 2+i/16:GOSUB 300
140 IF INKEY(79)<>-1 THEN PLOT XPOS,YPOS,0:t=t-1:DRAW a(t,0),a(t,1):
PLOT a(t,0),a(t,1),1:GOTO 180
160 IF INKEY(58)<>-1 GOTO 400
170 IF t>199 THEN PRINT"Stop":n=200:GOTO 500
180 LOCATE 1,1:PRINT USING "###";t
190 LOCATE 1,25:PRINT XPOS;:PRINT"/"YPOS;
200 GOTO 80
300 '====================================================== Placement du point
310 PLOT XPOS,YPOS,h
320 h=TEST(XPOS+u,YPOS+v)
330 PLOT XPOS+u,YPOS+v,1
340 RETURN
400 '=============================================== Enregistrement du point
410 tt=t MOD(2)
420 IF tt=0 THEN PLOT XPOS,YPOS,2:a(t,0)=XPOS:a(t,1)=YPOS:t=t+1:PRINT
CHR$(7):GOTO 180
430 a(t,0)=XPOS:a(t,1)=YPOS:PLOT a(t-1,0),a(t-1,1),2:DRAW a(t,0),a(t,1):
t=t+1:PRINT CHR$(7):GOTO 180
500 '========================================================== Initialisation
510 PLOT 0,0,1:CLS:DEG
520 INPUT "Hauteur des murs ";HH
530 MODE 1:INK 0,0:BORDER 0:PEN 1:ORIGIN 320,200
540 m=n/2-1:theta=45:psi=45
550 DIM b%(m,4)
560 DIM p%(4*m,1)
570 DIM q%(m,2)
600 '================================================ PREPARATION DU DESSIN
610 c1=COS(psi):s1=SIN(psi)
620 c2=COS(theta):s2=SIN(theta)
630 h=INT(HH*s2):j=0
640 FOR t=0 TO m
650 u=INT(s1*a(2*t,0)-c1*a(2*t,1))
660 v=INT(s1*a(2*t+1,0)-c1*a(2*t+1,1))
670 IF u>v THEN b%(t,0)=v:b%(t,1)=u ELSE b%(t,0)=u:b%(t,1)=v
680 u1=INT(s2*HH-c2*(c1*a(2*t,0)+s1*a(2*t,1)))
690 v1=INT(s2*HH-c2*(c1*a(2*t+1,0)+s1*a(2*t+1,1)))
700 IF u>v THEN b%(t,2)=v1:b%(t,3)=u1 ELSE b%(t,2)=u1:b%(t,3)=v1
710 NEXT
720 FOR t=0 TO m
730 GOTO 800
740 NEXT t
750 GOTO 3000
800 '=============================================== Intersections d'un mur
810 k=0:a=b%(t,0):b=b%(t,2)-h:c=b%(t,1):d=b%(t,3)-h:V0=c-a:V1=d-b
820 FOR w=0 TO 9:q%(w,2)=0:q%(w,0)=-1000:NEXT
900 '================== intersections partie basse
910 FOR u=0 TO m
920 quot=V0*(b%(u,3)-b%(u,2))-V1*(b%(u,1)-b%(u,0))
930 IF quot=0 GOTO 960
940 lambda = ((a-b%(u,0))*(b%(u,3)-b%(u,2))-(b-b%(u,2))*(b%(u,1)-b%(u,0)))/
quot
950 IF lambda<0 AND lambda>-1 THEN GOSUB 1400
960 IF V0=0 GOTO 990
970 x=b%(u,0):y0=b%(u,2):f9=-1:GOSUB 1300
980 x=b%(u,1):y0=b%(u,3):f9=1:GOSUB 1300
990 NEXT u
```

```
1000 b%(t,4)=k:IF k=0 THEN b%(t,4)=100:GOTO 1100
1010 fg=0:fgs=0
1020 FOR v=0 TO k-1
1030 IF fg*q%(v,2)>0 THEN b%(t,4)=b%(t,4)-1:IF fg=-1 GOTO 1080 ELSE j=j-1:
GOTO 1050
1040 IF v>0 THEN IF q%(v,0)=q%(v-1,0) THEN j=j-2+fgs:b%(t,4)=b%(t,4)-2+fgs:
fgs=1:fg=-1:GOTO 1070 ELSE fgs=0
1050 p%(j,0)=q%(v,0):p%(j,1)=q%(v,1)
1060 fg=q%(v,2)
1070 j=j+1
1080 NEXT
1090 IF b%(t,4)=0 THEN b%(t,4)=100
1100 '==================== inters. 1ere verticale
1110 x=a:y=b
1120 FOR u=0 TO m
1130 IF x<=b%(u,0) OR x>=b%(u,1) GOTO 1160
1140 y0=y:y=b%(u,2)+(a-b%(u,0))*(b%(u,3)-b%(u,2))/(b%(u,1)-b%(u,0))
1150 IF y>=b+h OR y<=y0 THEN y=y0
1160 NEXT:IF y>b GOTO 1450
1170 FOR u=0 TO m
1180 IF b+h=b%(u,2) AND a=b%(u,0) AND u<>t GOTO 1210
1190 NEXT
1200 GOTO 1600
1210 IF V1>(b%(u,3)-b%(u,2)) THEN b%(t,4)=b%(t,4)+10
1220 GOTO 1600
1300 '==================== inters. avec verticale
1310 IF x<=a OR x>=c THEN RETURN
1320 y=INT(b+(x-a)*V1/V0)
1330 IF y>=y0 OR y<=y0-h THEN RETURN
1340 GOTO 1500
1400 '====================== inters. avec oblique
1410 x=INT(a-lambda*V0):IF x<=b%(u,0) OR x>=b%(u,1) THEN RETURN
1420 y=INT(b-lambda*V1)
1430 IF V1<(b%(u,3)-b%(u,2)) THEN fg=-1 ELSE fg=1
1440 GOTO 1500
1450 '================= inters. avec partie haute
1460 b%(t,4)=-b%(t,4)
1470 p%(j,0)=x:p%(j,1)=y:j=j+1
1480 p%(j,0)=c:p%(j,1)=d:j=j+1
1490 GOTO 1600
1500 '================================= Enregistrement dans pile secondaire
1510 IF x<q%(0,0) OR k=0 THEN vv=0:GOTO 1550
1520 vv=k:FOR v=0 TO k-1
1530 IF x<q%(v+1,0) AND x>=q%(v,0) THEN vv=v+1:GOTO 1550
1540 NEXT
1550 FOR v=k TO vv+1 STEP -1
1560 q%(v,0)=q%(v-1,0):q%(v,1)=q%(v-1,1):q%(v,2)=q%(v-1,2)
1570 NEXT
1580 q%(vv,0)=x:q%(vv,1)=y:q%(vv,2)=fg
1590 k=k+1:RETURN
1600 '==================== inters. 2eme verticale
1610 x=c:y=d
1620 FOR u=0 TO m
1630 IF x<=b%(u,0) OR x>=b%(u,1) GOTO 1660
1640 y0=y:y=b%(u,2)+(c-b%(u,0))*(b%(u,3)-b%(u,2))/(b%(u,1)-b%(u,0))
1650 IF y>=d+h OR y<=y0 THEN y=y0
1660 NEXT
1670 IF y<=d GOTO 740
1680 IF b%(t,4)<0 THEN p%(j-1,0)=x:p%(j-1,1)=y:GOTO 740
1690 b%(t,4)=-b%(t,4)
1700 p%(j,0)=a:p%(j,1)=b:j=j+1
1710 p%(j,0)=x:p%(j,1)=y:j=j+1
1720 GOTO 740
1900 '========================================== Test des touches reunies
1910 IF a=0 THEN b=10
1920 IF a=32 THEN b=20
1930 IF a=128 THEN b=5
1940 IF a=160 THEN b=45
1950 RETURN
2000 '============================================== Touches de rotation
2010 a$=INKEY$:IF a$="" GOTO 2010
2020 a=INKEY(34):IF a<>-1 THEN GOSUB 1900:psi=psi-b:GOTO 600
2030 a=INKEY(27):IF a<>-1 THEN GOSUB 1900:psi=psi+b:GOTO 600
2040 a=INKEY(67):IF a<>-1 THEN GOSUB 1900:theta=theta-b:GOTO 600
```

```
2050 a=INKEY(59):IF a<>-1 THEN GOSUB 1900:theta=theta+b:GOTO 600
2060 IF INKEY(79)<>-1 THEN CLS:END
2070 GOTO 2010
3000 '=============================================================== DESSIN
3010 CLS:j=0
3020 FOR t=0 TO m
3030 K0=ABS(b%(t,4)):k=K0 MOD(10)
3040 IF K0=100 THEN K0=0
3050 PLOT b%(t,1),b%(t,3):DRAW b%(t,0),b%(t,2)
3060 IF b%(t,4)=-100 THEN DRAW p%(j,0),p%(j,1):MOVE p%(j+1,0),p%(j+1,1):
DRAW b%(t,1),b%(t,3):j=j+2:GOTO 3140
3070 IF b%(t,4)=-110 THEN j=j+2:PLOT p%(j-1,0),p%(j-1,1):
DRAW b%(t,1),b%(t,3):GOTO 3140
3080 IF K0=110 GOTO 3110
3090 IF b%(t,4)<0 GOTO 3160
3100 IF k<>K0 THEN f9=1:MOVE p%(j,0),p%(j,1):GOTO 3300
3110 DRAWR 0,-h
3120 IF k<>0 THEN f9=-1:DRAW p%(j,0),p%(j,1):GOTO 3300
3130 DRAW b%(t,1),b%(t,3)-h:DRAWR 0,h
3140 NEXT t
3150 GOTO 2000
3160 DRAW p%(j+k,0),p%(j+k,1)
3170 IF b%(t,2)=p%(j+k,1)+h AND K0=k THEN f9=-1:DRAW p%(j,0),p%(j,1):
GOTO 3300
3180 f9=1:MOVE p%(j,0),p%(j,1):GOTO 3300
3300 '============================================== Dessin parties basses
3310 i=0
3320 i=i+1:IF i=k GOTO 3400
3330 IF f9=1 THEN DRAW p%(j+i,0),p%(j+i,1) ELSE MOVE p%(j+i,0),p%(j+i,1)
3340 f9=-f9
3350 GOTO 3320
3400 '========================================================= Fin du mur
3410 IF b%(t,4)<0 GOTO 3450
3420 j=j+k
3430 IF f9=1 THEN DRAW b%(t,1),b%(t,3)-h:DRAW b%(t,1),b%(t,3)
3440 GOTO 3140
3450 j=j+k+2
3460 IF f9=1 THEN DRAW  p%(j-1,0),p%(j-1,1) ELSE PLOT p%(j-1,0),p%(j-1,1)
3470 DRAW b%(t,1),b%(t,3):GOTO 3140
```

Tout d'abord, le programme travaille mur après mur. Rappelons que la matrice *a* contient les segments qui forment la vue en plan de l'appartement. Vous constaterez en effet que, au cours de l'utilisation du plot baladeur, seuls des segments sont enregistrés, en ce sens que si le nombre affiché en haut (*t*) est pair, le point enregistré (par la touche *e*) sera isolé, sinon il sera rejoint au précédent. Aussi n'oubliez pas, si vous souhaitez faire deux segments jointifs, d'enregistrer deux fois le point de jonction. J'aurai de toute façon l'occasion de donner d'autres conseils de bonne utilisation de cette première partie du programme.

Une fois votre plan engrangé dans la matrice *a*, le programme demande la hauteur des murs HH. (Donnez une valeur importante, par exemple 60.) Cela fait, on passe à l'alternance : attente d'une touche, dessin, attente d'une touche, etc., comme pour les polyèdres. De même, les deux angles psi et thêta sont initialisés à 45 degrés.

Soit un segment de la matrice a. Le programme, au cours de ses calculs (lignes 600 à 1800), va transformer ce segment, comme les autres, en un mur. Soit t le numéro de ce segment. Il s'agit du segment joignant les points n[os] $2t$ et $2t+1$ de la matrice a. Soit R et S les points projection des points initiaux du segment sur l'écran, et P et Q la projection des points de coordonnées $(a(2t,0), a(2t,1), \mathrm{HH})$ et idem avec $2t+1$ pour Q. L'axe Oz étant vertical (c'est ici essentiel), R est en dessous de P, et S en dessous de Q. La distance entre R et P, ou entre S et Q, est $h = \mathrm{HH} \cdot s_2$. Cette valeur h représente la hauteur de tous les murs en projection sur l'écran. C'est donc une constante, tant que les angles n'ont pas changé. A présent, soit $b(t,0), b(t,2)$ les coordonnées projetées de P, et $b(t,1)$, $b(t,3)$ celles de Q. Les coordonnées (a,b) de R et (c,d) de S s'en déduisent aisément : $a = b(t,0)$ et $c = b(t,1)$ (points sur la même verticale), et $b = b(t,2) - h$, $d = b(t,3) - h$ (... à une distance h). On a ainsi assez facilement les coordonnées des quatre coins du mur t.

Je dois préciser que, pour simplifier les calculs d'une part, pour ne pas alourdir la mémoire d'autre part, toutes ces matrices sont à coefficients entiers, même h est pris entier, et ainsi de tout ce qui sera calculé par la suite.

En ligne 600 et suivantes, on remplit donc les quatre premières colonnes de la matrice b (pour la cinquième, nous y reviendrons), mais avec une précision importante : en mettant toujours en premier le point dont l'abscisse est minimale. Pour toute valeur de t, $b(t,0)$ est inférieur à $b(t,1)$ (donc a est inférieur à c). Ce, non par caprice, mais par nécessité.

Ces calculs faits, on passe à l'étude mur après mur (lignes 800 à 1800). Soit un mur PQRS comme celui de la Figure VII.B, schéma a. On voit que la partie basse de ce mur est interrompue en trois points, I, J, K, et que, des deux côtés, un seul est interrompu par une autre face, c'est le côté QS, coupé en L. Ce sont ces points I, J, K, L qu'il s'agit de trouver, puis de calculer et enfin de stocker afin de les restituer. Cette restitution est bien sûr impossible si l'ordinateur ne sait pas d'où viennent ces points ni de quel mur ils proviennent. D'où le rôle de l'indicateur $b(t,4)$ qui, pour chaque mur, renseigne sur les points suivants :

- Par son signe, sur l'état des verticales ; s'il est positif, les deux verticales sont entières, sinon l'une au moins est coupée.
- Par sa valeur absolue, qui est un nombre entier d'au plus trois chiffres ; si elle est supérieure à 100, c'est que la partie basse RS n'est pas interrompue ; si elle est supérieure à 10 (chiffre des dizaines mis, en fait),

c'est que la première verticale PR est commune à un autre mur, sinon cette verticale est isolée.

- Enfin, le chiffre des unités indique le nombre de points situés sur RS, et où il y a interruption (c'est-à-dire où l'on passe du visible à l'invisible, ou l'inverse).

Dans l'exemple de la Figure VII.B.*a,* cet indicateur vaut – 3. En effet :

- – signe négatif, car la seconde verticale est interrompue.
- Valeur absolue 003 car :

  RS est interrompue (d'où le premier 0).

  La première verticale PR n'appartient pas à un autre mur (d'où le second 0).

  Enfin, 3 est le nombre de points d'intersection de la partie basse (I, J, K).

Je reconnais que ce codage n'est pas d'une folle simplicité, mais il évite d'utiliser plusieurs flags ou indicateurs, ce qui rallongerait démesurément la matrice *b* et risquerait d'engendrer un *overflow*. Il faudrait exactement deux flags et un indicateur, soit deux colonnes supplémentaires.

Cet indicateur renseigne sur l'état du mur. Quant aux coordonnées des points concernés, autres que P, Q, R, S, elles sont stockées dans l'ordre dans la matrice *p* (qui joue le rôle de pile). Par exemple, lors de la lecture de l'état du mur, le compteur *j* aura une certaine valeur ; chaque fois que le programme de traçage (lignes 3000 et suivantes) aura été chercher un point dans la pile, il augmentera *j* pour que ce compteur *pointe* sur le point suivant.

Reste à savoir comment ces points sont trouvés, et comment l'indicateur sera positionné.

Tout d'abord, de la ligne 900 à la ligne 1100, le programme cherche les intersections de la partie basse ; d'une part avec toutes les autres parties basses (lignes 920 à 950), puis avec les premières (ligne 970) et dernières (ligne 980) verticales. Peu importe en réalité le détail des calculs ; nous ne nous y attarderons pas car c'est un simple calcul d'intersection de droites, avec vérification (les points d'intersection doivent se trouver sur les segments voulus). Au fur et à mesure que ces points d'intersection sont calculés, ils sont stockés sur une petite matrice (pile secondaire *q*), mais *classés* par ordre d'abscisses croissantes. En outre, la dernière colonne de *q* contient un flag fg. Ce flag est à – 1 si le point est de type *rentrant*, à + 1 s'il est de type *sortant*. Dans le premier cas, le trait s'arrête au point et

rentre derrière l'autre mur, donc est caché, c'est le cas des points I et K sur la Figure VII.B.*a*. Dans le second cas, le trait semble sortir de derrière l'autre mur, comme pour le point J.

Lorsque toutes les autres faces ont été examinées, on va faire le tri dans cette pile secondaire avant de replacer les points sur la pile principale. En effet, examinez par exemple sur la Figure VII.B, schéma 2 du *b*, la situation du point J. Si le plan 2 n'existait pas, J serait un sortant. Mais, à cause du plan 2, J devient un faux sortant ; le trait reste en pointillé après y être passé. Le vrai sortant est ici K. Les points étant classés par ordre d'abscisses croissantes dans *q*, J se trouve avant K ; or, ce point J n'a aucun intérêt, aucun trait visible n'y aboutit, il doit donc être éliminé. C'est ce qui se passe dans les lignes 1000 à 1080, lors du transfert d'une pile à l'autre. La ligne 1030 en effet exclut qu'il y ait deux points successifs ayant la même nature, et ce, de la façon suivante : si deux sortants se suivent, on garde le second ; si deux rentrants se suivent, on garde le premier.

Dans cette partie essentielle du programme, on exclut également les points multiples. En effet, toujours sur ce même schéma, en b_2, le point I est un point double ; appartenant à deux faces, il va se trouver deux fois sur la pile provisoire. Il n'y aura pas de problème, car il s'y trouvera deux fois avec le même flag (deux fois rentrant ici). L'un des deux sera donc éliminé, et I ne sera stocké qu'*une* fois sur la pile principale *p*.

Mais que se passera-t-il dans un cas plus délicat, comme par exemple celui du schéma b_3 ? En effet, on trouvera sur la pile secondaire les points suivants : I (*r*), J (*s*), J (*r*), J (*r*), K (*s*), L (*s*) ; (*r*) signifiant rentrant, et (*s*) sortant. En effet, J appartient aux trois murs de devant, est sortant pour le mur 1, mais rentrant pour les deux autres. K et L étant de même nature, tout comme les deux derniers J, un premier dégrèvement mettrait dans la pile les points suivants : I, J, J, L. Mais ces deux J ne servent à rien car, ici encore, aucun trait du mur grisé PQRS n'arrive jusqu'à J. C'est pourquoi la ligne 1040 se charge d'éliminer ces deux points indésirables. Tous les points multiples seront ainsi *gommés*. Ainsi, l'encombrement de la pile principale sera réduit au minimum, et le tracé final plus rapide.

Au total, on comprend pourquoi les trois schémas b_1, b_2, b_3 de la Figure VII.B aboutissent tous au même indicateur : 2 (marqué en dessous). Chaque fois, des points indésirables sont ôtés, et c'est pourquoi ces cas de figure d'apparences si différentes (mais pas du point de vue du mur PQRS, qui reste toujours avec deux coupures) aboutissent finalement au même résultat, à partir d'une pile provisoire très variable.

Ensuite, le programme cherche les intersections de la première verticale avec les autres murs. Deux cas très différents sont étudiés (lignes 1100 à 1300). Tout d'abord, la première verticale peut être franchement coupée, comme l'est QS sur le schéma *a*. Dans ce cas, rien de bien spécial. Toutefois, notez que là encore on n'enregistre, de tous les points de coupe sur RP, que le plus haut (le plus proche de P), les autres n'ayant aucune importance. L'indicateur est alors positionné négatif, tout comme lorsqu'on trouve un point d'intersection sur la seconde verticale (lignes 1600 à 1700). Dans ces cas, le point d'intersection sur la première verticale, puis le point d'intersection sur la seconde verticale sont placés sur la pile *p*, à la suite des autres. Si l'une des deux verticales n'est pas coupée, on prend comme point d'intersection son point le plus bas (par exemple S pour la seconde).

Mais, pour la première verticale, et pour elle seulement, un cas très spécial peut se produire, c'est le cas visible en b_4 : PR est aussi la première verticale d'un autre mur, le mur 1. C'est ici que le classement par ordre des verticales est essentiel, car si PR était la seconde verticale d'un autre mur, cela n'aurait aucune importance (cas des murs 1 et 2 sur b_3). Par contre, ici, le programme risque de faire erreur. En effet, dans le cas du schéma b_4, si l'indicateur est mis à 1 (un seul point sur RS), le programme va tracer de P à R et de R à I, ce qui ne doit pas être. Pour l'éviter, on ajoute dix à l'indicateur, et le programme de restitution, en ligne 3000, se charge du reste. Notons toutefois que cet ajout de dix exige un test (ligne 1210). En effet, en b_5, le mur 1 a aussi PR comme première verticale, mais il est derrière, donc le mur PQRS est complètement visible (indicateur à 100). Ce test porte sur le caractère plus ou moins incliné de l'arête inférieure (valeur de V_1), la plus dressée vers le haut étant celle qui est invisible. Le schéma b_6 donne encore un exemple différent de ce type de cas.

Ainsi, nous voyons qu'après les calculs la valeur de l'indicateur et la pile vont permettre de tracer chaque mur correctement. Nous n'avons pas fait ici de grande théorie, mais uniquement de la pratique, basée sur une bonne observation des choses (et encore, on peut sûrement faire encore bien mieux). Mais la contrepartie en est que l'on doit traiter de nombreux cas particuliers (voyez la partie “Dessin”), ce qui n'éclaire pas tellement les choses.

L'essentiel est bien sûr que le programme marche, sous quelques réserves toutefois. La première fut déjà dite : l'appartement doit être vu par le haut. Si thêta devient supérieur à 90, le tracé sera erroné. D'autre

part, les deux piles sont limitées. Pour la pile principale, ce ne sera guère gênant, je pense, mais cela peut l'être pour la secondaire. Il peut alors y avoir quelques problèmes. Dans ce cas, une bonne solution : raccourcissez vos murs. Préférez à un très grand mur deux ou trois petits les uns à côté des autres. De même, ne faites pas partir un mur du milieu d'un autre. Ainsi, si vous avez un T dans votre plan, il faut enregistrer *trois* murs : la barre verticale, la partie gauche de la barre horizontale, et la partie droite. Si vous enregistrez la barre horizontale en un seul morceau, la verticale partira de son milieu, et il se produira des erreurs. D'une façon générale, en cas d'erreur, subdivisez vos murs ; le tracé sera plus long mais exact.

On pourrait encore, en travaillant sur quatre niveaux au lieu de deux, créer des portes et des fenêtres ; le principe général resterait le même. Enfin, le même genre de méthode s'applique pour les polyèdres concaves ; pour chaque arête, il faut calculer toutes ses intersections avec les autres : la longueur du programme, en lignes comme en durée d'exécution, serait telle qu'il n'en vaut vraiment pas la peine. Là aussi, il faudra s'adapter à l'usage que l'on peut faire de ces polyèdres... ou utiliser un autre langage, plus rapide que le BASIC.

3. LES FACES BOMBEES : SURFACES POLYNOMIALES — PRINCIPE DU TRACE D'HORIZON — BORDURES

Jusqu'à présent, nous avons essentiellement parlé de plans, pour toutes les surfaces que nous avons étudiées, ou de morceaux de plans (faces des polyèdres, murs...).

Cependant, il va sans dire qu'il n'y a pas que des surfaces planes dans l'univers, bien au contraire.

Les surfaces bombées peuvent, bien entendu, être représentées d'une manière approximative par des polyèdres aux très petites faces, mais la chose n'est guère élégante, car la surface est alors parsemée de traits qui n'existent pas en réalité. Or certaines surfaces s'en accommodent très mal.

Il est possible de représenter des surfaces bombées sur un ordinateur, de deux manières différentes : avec, ou sans horizon. La seconde manière sera exposée au Chapitre IX ; nous allons traiter ici de la première.

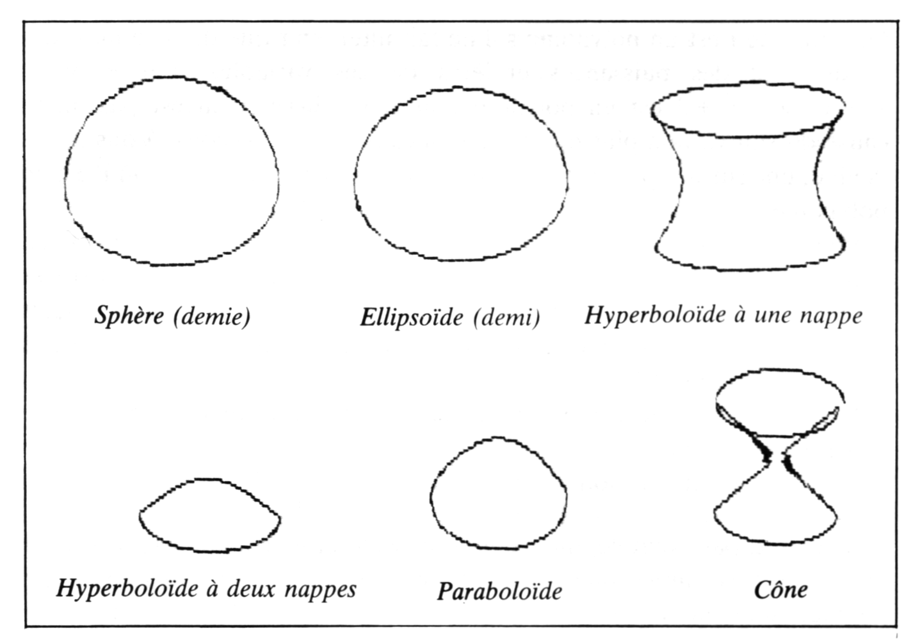


Figure VII.C

Comprendre ce qu'est l'horizon d'une surface n'est guère difficile. Lorsque vous regardez un ballon de rugby, par exemple (ellipsoïde en mathématiques), vous en voyez la bordure ; au-delà de celle-ci, le sol, vos mains, l'environnement en un mot, sont visibles. A l'intérieur de cette bordure, vous ne voyez que le ballon, qui masque le paysage. Cette bordure, nous l'appellerons horizon, car l'horizon (au sens usuel) n'est jamais que l'équivalent de cette bordure, le ballon étant remplacé par notre planète, la Terre.

Dans un cas comme dans l'autre, l'horizon est une entité artificielle, en ce sens qu'il s'agit d'une ligne imaginaire qui dépend, comme chacun sait, de la position où l'on se trouve. Néanmoins, cet horizon peut être tracé par votre ordinateur, pour certaines surfaces dites polynomiales. On peut également tracer l'intersection d'une de ces surfaces avec un plan qui le limite ; on peut ainsi tracer également une demi-sphère, etc. Le tout sous n'importe quel point de vue.

Qu'est-ce que ces surfaces polynomiales ? Ce n'est pas compliqué non plus. D'une façon générale, une surface est donnée par une équation du type $f(x, y, z) = 0$, f étant une fonction donnée des trois variables x, y, z.

On dira que f est un polynôme s'il ne fait intervenir que des sommes, des produits et des puissances entières de ces variables. Par exemple $x^5 + x \cdot y^2 \cdot z^7 + 3$ est un polynôme, mais $x \cdot \sin(y)$ n'en est pas un (à cause du sinus), non plus que $z \cdot y/x$ (à cause de la division). Vous l'avez deviné, une surface polynomiale a pour équation $P(x, y, z) = 0$, où P est un polynôme.

Pourquoi ces surfaces-là plutôt que d'autres (qui seront vues au Chapitre IX) ? Parce que ce sont les seules pour lesquelles on peut tracer simplement l'horizon. Toutes les autres exigent de très complexes calculs de dérivation, qui sortent complètement du cadre de cet ouvrage. (Voyez à ce sujet l'Annexe B.)

Mais, me direz-vous, si je ne peux pas tracer n'importe quelle surface, je suis forcément limité.

Eh bien, justement non.

Imaginez, par exemple, que vous vouliez tracer un bateau. Même si c'est un bateau très simple, une barque, vous pourrez chercher longtemps une fonction donnant l'équation de la surface de la coque. Même en admettant que vous la trouviez, son seul énoncé complet serait si long que votre Amstrad risque alors d'être le premier ordinateur de l'histoire à se mettre en grève.

Par contre, une barque peut être décomposée en gros en trois morceaux de coque : une paroi droite à l'arrière, le côté gauche (bombé), et le côté droit similaire, ces deux derniers se joignant au niveau de l'étrave, sur le plan de symétrie de la barque. Ces deux morceaux bombés peuvent sans difficultés être considérés comme des surfaces polynomiales (car ils sont de forme assez simple) limitées par deux plans : le plan de symétrie du bateau, et le plan de la paroi arrière. En traçant successivement ces deux morceaux de surfaces bombées, l'un contre l'autre, puis le bord de la partie arrière, vous obtiendrez ainsi votre barque d'une façon très réaliste.

Or, de tels tracés sont possibles. Pas directement par le Programme VII.2 qui ne peut tracer toutes les surfaces polynomiales. En effet, une surface polynomiale est caractérisée, comme son polynôme d'équation, par son degré (définition précise plus loin). Ce degré traduit, en gros, la plus ou moins grande complexité de la surface. Le degré 1 ne représente que les plans. Le degré 2, que seul permet le Programme VII.2, admet déjà un grand nombre de surfaces très variées (dont des exemples seront donnés en fin de chapitre). Naturellement, plus le degré est important, plus le programme correspondant doit être complexe, comme

nous le verrons. C'est pourquoi j'ai tenu à ne pas trop compliquer les choses avec ce programme, mais j'expliquerai néanmoins comment faire mieux, surtout dans l'Annexe B.

Avant de continuer, je dois encore parler de l'utilité de ces surfaces polynomiales.

Nous avons étudié le tracé des polyèdres bien profondément. L'inconvénient des polyèdres tient en une phrase : leur faces sont plates, donc ils ne *collent* pas très bien à la réalité.

Qu'à cela ne tienne. Nous leur donnerons des faces bombées. Vous devinez de quel type : polynomial... D'abord de degré 2 puis, si cela se révèle nécessaire, de degré supérieur, suivant l'objet à imiter.

Je ne donne pas là les intentions de cet ouvrage, car j'en outrepasserais alors le cadre, mais celles qui furent probablement les premières des inventeurs de la C.A.O. (Conception Assistée par Ordinateur). En effet, c'est ainsi que les ordinateurs puissants destinés à cet usage créent avions, navires, automobiles sur leurs écrans, images somptueuses dont vous avez probablement vu des exemples à la télévision ou dans des magazines.

Cette création, si vous assimilez ce qui va suivre et si vous avez bien compris les problèmes posés par les polyèdres, vous sera tout à fait possible à condition que vous soyez courageux, patient, et un bon programmeur d'assembleur car seul ce langage pourrait permettre une vitesse correcte d'exécution pour des objets compliqués !

Avant d'en arriver à ces multiplicités de faces bombées, essayons déjà d'en tracer une seule, cela suffira à notre plaisir.

Expliquons déjà ce qu'est le degré. Un polynôme de trois variables, formant l'équation d'une surface polynomiale, est la somme d'un certain nombre de monômes, du type $a \cdot x^u \cdot y^v \cdot z^w$, où a est une constante quelconque, et u, v, w trois entiers positifs. La somme $s = u + v + w$ de ces trois entiers est appelée degré du monôme. Le degré du polynôme est le plus grand de tous les degrés des monômes qui le constituent. Par exemple, si le polynôme est

$$P\,(x, y, z) = 4 \cdot x \cdot y^2 - y \cdot z \cdot x^3 + \frac{z}{8}$$

nous y voyons trois monômes :

$$4 \cdot x \cdot y^2 ; \quad -y \cdot z \cdot x^3 ; \quad \frac{z}{8}$$

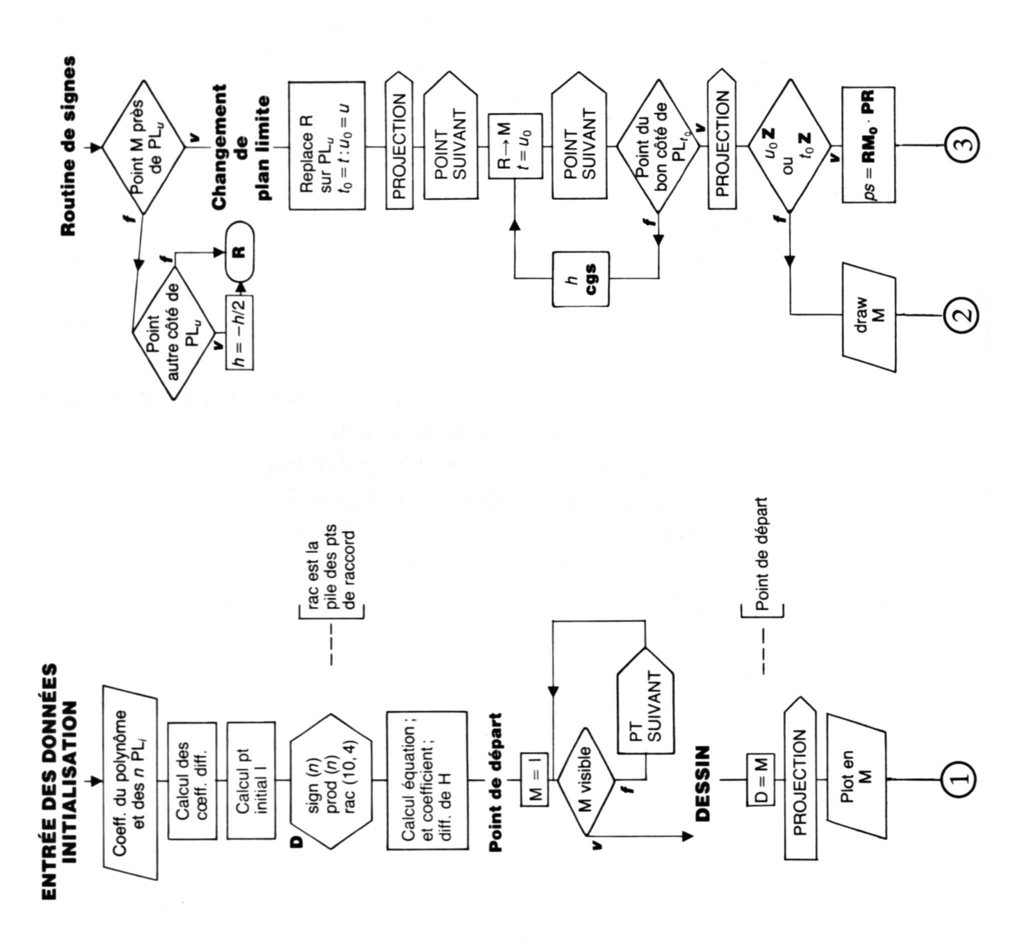
Dans le premier, on a : $a = 4$; $u = 1$; $v = 2$; $w = 0$ (un nombre à la puissance zéro vaut 1), donc $s = 3$. Dans le deuxième : $a = -1$;

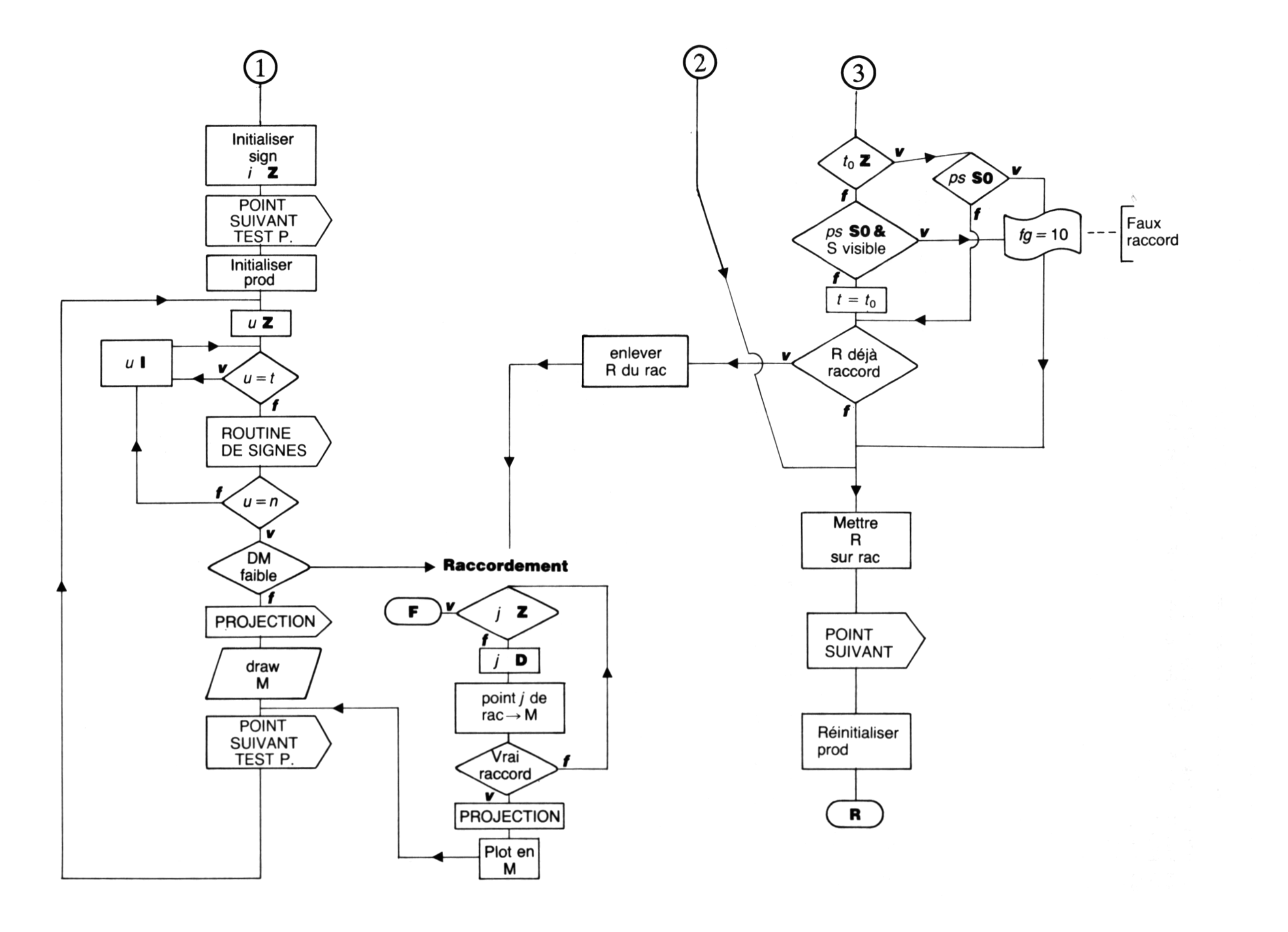
$u = 3$; $v = 1$; $w = 1$, donc $s = 5$. Dans le dernier : $a = 1/8 = 0{,}125$; $u = 0$; $v = 0$; $w = 1$; donc $s = 1$. La plus grande valeur de s est obtenue pour le deuxième monôme, où $s = 5$. Nous dirons que P est un polynôme de degré 5.

Organigramme VII.2

SURFACES POLYNOMIALES

Calcul de S





1
Initialiser sign i Z
POINT SUIVANT TEST P.
Initialiser prod
u Z
u = t
u I
ROUTINE DE SIGNES
u = n
DM faible
Raccordement
PROJECTION
draw M
POINT SUIVANT TEST P.
F
j Z
j D
point j de rac → M
Vrai raccord
PROJECTION
Plot en M
2
3
t0 Z
ps S0
ps S0 & S visible
fg = 10
Faux raccord
t = t0
R déjà raccord
enlever R du rac
Mettre R sur rac
POINT SUIVANT
Réinitialiser prod
R
v
f

PROGRAMME VII.2

```
10 '======================================= SURFACES POLYNOMIALE DE DEGRE 2

20 '===================================== ENTREE DES DONNEES.INITIALISATION

30 MODE 1:INK 0,0:BORDER 0:INK 1,24:INK 2,6:PEN 2
40 PRINT TAB(5),"ENTREE DU POLYNOME":PRINT:PRINT
50 PEN 1:DIM p(9)
60 INPUT "Terme constant ";p(0):PRINT
70 INPUT "Terme en x ";p(1):PRINT
80 INPUT "Terme en y ";p(2):PRINT
90 INPUT "Terme en z ";p(3):PRINT
100 INPUT "Terme en x*x ";p(4):PRINT
110 INPUT "Terme en y*y ";p(5):PRINT
120 INPUT "Terme en z*z ";p(6):PRINT
130 INPUT "Terme en x*y ";p(7):PRINT
140 INPUT "Terme en x*z ";p(8):PRINT
150 INPUT "Terme en y*z ";p(9):PRINT
160 CLS:PEN 2
170 PRINT TAB(5),"PLANS LIMITES":PRINT:PRINT
180 PEN 1:INPUT "Nombre de plans limites ";n
190 PRINT:PRINT"Entrez les coefficients des equations""des plans limites
sous la forme :""u*x+v*y+w*z+k=0"
200 PRINT:DIM pl(n,3)
210 FOR t=1 TO n
220 PRINT:PRINT"Equation du plan numero "t:PRINT:PRINT
230 INPUT;"u ";u:INPUT;": v ";v:INPUT;": w ";w:INPUT;": k ";k
240 r=SQR(u*u+v*v+w*w):pl(t,0)=k/r:pl(t,1)=u/r:pl(t,2)=v/r:pl(t,3)=w/r
250 NEXT
260 CLS:INPUT "Precision du trace ";h
270 CLS:DEG
300 '============================== Calcul des coefficients differentiels
310 DIM q(n,5)
320 FOR t=1 TO n
330 d=-pl(t,3)
340 q(t,0)=p(0)+pl(t,0)*(p(3)+p(6)*pl(t,0)/d)/d
350 q(t,1)=p(1)+pl(t,1)*(p(3)+2*pl(t,0)*p(6)/d)/d+pl(t,0)*p(8)/d
360 q(t,2)=p(2)+pl(t,2)*(p(3)+2*pl(t,0)*p(6)/d)/d+pl(t,0)*p(9)/d
370 q(t,3)=p(4)+pl(t,1)*(p(8)+p(6)*pl(t,1)/d)/d
380 q(t,4)=p(5)+pl(t,2)*(p(9)+p(6)*pl(t,2)/d)/d
390 q(t,5)=p(7)+pl(t,2)*(p(8)+2*pl(t,1)*p(6)/d)/d+pl(t,1)*p(9)/d
400 IF flag=1 THEN RETURN ELSE NEXT
500 '============================================ Calcul du point initial
510 c=q(1,0):a=q(1,1):b=q(1,2):u=q(1,3):v=q(1,4):w=q(1,5)
520 aa=w*w-4*v*u:bb=w*b-2*v*a:cc=b*b-4*v*c
530 IF cc>=0 THEN xi=0:GOTO 560
540 IF aa>=0 THEN xi=-bb/aa:GOTO 560
550 xi=2*ABS(bb/aa)
560 IF v=0 THEN yi=-(c+xi*(a+xi*u))/(w*xi+b):GOTO 590
570 IF w*xi+b=0 THEN yi=SQR(-(c+xi*(a+xi*u))/v):GOTO 590
580 yi=(SQR((w*xi+b)^2-4*v*(c+xi*(a+xi*u)))-(w*xi+b))/v/2
590 zi=-(pl(1,0)+xi*pl(1,1)+yi*pl(1,2))/pl(1,3)
600 INPUT "Echelle ";e:CLS
610 DIM sign(n):DIM prod(n)
620 DIM rac(10,4)
700 '==================================== Angles et coefficients d'horizon
710 INPUT "Angle psi ";psi
720 PRINT:PRINT:INPUT "Angle theta ";theta
730 CLS
740 n0=COS(psi)*SIN(theta)
750 n1=SIN(psi)*SIN(theta)
760 n2=COS(theta)
770 pl(0,3)=n0*p(8)+n1*p(9)+2*n2*p(6):IF pl(0,3)<0.001 THEN pl(0,3)=0.001
780 pl(0,0)=n0*p(1)+n1*p(2)+n2*p(3)
790 pl(0,1)=2*n0*p(4)+n1*p(7)+n2*p(8)
800 pl(0,2)=n0*p(7)+2*n1*p(5)+n2*p(9)
810 t=0:flag=1:GOSUB 330
900 '======================================= Recherche du point de depart
910 x=xi:y=yi:z=zi:GOSUB 1200
920 t=1:h=10*h
930 IF indic=1 GOTO 1000
940 GOSUB 1300:GOSUB 2500
```

```
950 GOSUB 1200:IF indic=1 GOTO 1000
960 GOTO 940
1000 '======================================== DESSIN (Routine Principale)
1010 xd=x:yd=y:zd=z
1020 h=h/10:Prod0=1:divh=1
1030 MODE 2:INK 1,26:ORIGIN 320,200
1040 GOSUB 1400:PLOT XX,YY
1050 j=0
1060 FOR u=0 TO n:sign(u)=SGN(Pl(u,0)+Pl(u,1)*xi+Pl(u,2)*yi+Pl(u,3)*zi):
NEXT
1070 sign(1)=1
1080 GOSUB 1300:GOSUB 2500
1090 FOR u=0 TO n:Prod(u)=sign(u)*(Pl(u,0)+Pl(u,1)*x+Pl(u,2)*y+Pl(u,3)*z):
NEXT
1100 FOR u=0 TO n
1110 IF u<>t THEN GOSUB 1450
1120 NEXT
1130 IF (x-xd)*(x-xd)+(y-yd)*(y-yd)+(z-zd)*(z-zd)<h*h/2  GOTO 2000
1140 GOSUB 1400
1150 DRAW XX,YY
1160 GOSUB 1300:GOSUB 2500
1170 GOTO 1100
1200 '=================================================== Test de visibilite
1210 indic=SGN(Pl(0,0)+x*Pl(0,1)+y*Pl(0,2)+z*Pl(0,3))
1220 RETURN
1300 '============================================= Calcul du Point suivant
1310 x0=x:y0=y:z0=z
1320 dx=q(t,2)+2*y*q(t,4)+x*q(t,5)
1330 dy=-(q(t,1)+2*x*q(t,3)+y*q(t,5))
1340 dz=-(dx*Pl(t,1)+dy*Pl(t,2))/Pl(t,3)
1350 r=dx*dx+dy*dy+dz*dz:r=SQR(r)
1360 x=x+h*dx/r
1370 y=y+h*dy/r
1380 z=z+h*dz/r
1390 RETURN
1400 '============================== Coordonnees Projetees
1410 XX=(SIN(Psi)*x-COS(Psi)*y)*e
1420 YY=(SIN(theta)*z-COS(Psi)*COS(theta)*x-SIN(Psi)*COS(theta)*y)*e
1430 RETURN
1450 '==================================================== Routine de signes
1460 s=Pl(u,0)+Pl(u,1)*x+Pl(u,2)*y+Pl(u,3)*z
1470 Prod=s*sign(u)
1480 IF ABS(Prod)<ABS(h*divh/200) GOTO 1500
1490 IF Prod*Prod(u)>0 THEN Prod(u)=Prod:RETURN ELSE x=x0:y=y0:z=z0:h=-h/2:
divh=-2*divh:Prod(u)=Prod:RETURN
1500 '======================================== Changement de Plan limite
1510 XX0=XX:YY0=YY:t0=t:u0=u:f9=1:h=h*ABS(divh):divh=SGN(h)
1520 xc=x:yc=y:IF ABS(Pl(u,3))<0.01 THEN zc=z ELSE zc=-(Pl(u,0)+x*Pl(u,1)+
y*Pl(u,2))/Pl(u,3)
1530 GOSUB 1400:XXc=XX:YYc=YY
1540 GOSUB 1300:indic=SGN(Pl(u,0)+Pl(u,1)*x+Pl(u,2)*y+Pl(u,3)*z):s0=0
1550 t=u0:x=xc:y=yc:z=zc:GOSUB 1300
1560 u=t0:s=(Pl(u,0)+Pl(u,1)*x+Pl(u,2)*y+Pl(u,3)*z)*sign(t0)
1570 IF s<s0 THEN s0=s:h=-h:f9=-f9:x=xc:y=yc:z=zc:GOTO 1550
1580 GOSUB 1400
1590 IF u0<>0 AND t0<>0 THEN DRAW XX,YY:GOTO 1660
1600 Ps=(XXc-XX0)*(XX-XXc)+(YYc-YY0)*(YY-YYc)
1610 IF t0=0 GOTO 1690
1620 IF Ps>0 AND indic=-1 THEN x0=xc:y0=yc:z0=zc:f9=10:GOTO 1650
1630 t=t0:GOSUB 2100:IF f9rac>=0 GOTO 2200
1640 x=xc:y=yc:z=zc:h=f9*h
1650 rac(j,0)=xc:rac(j,1)=yc:rac(j,2)=zc:rac(j,3)=f9:rac(j,4)=u0:j=j+1
1660 GOSUB 1300
1670 FOR u=0 TO n:Prod(u)=sign(u)*(Pl(u,0)+Pl(u,1)*x+Pl(u,2)*y+Pl(u,3)*z):
NEXT
1680 RETURN
1690 IF Ps>0 THEN indic=-1:GOTO 1620
1700 t=t0:GOSUB 2100:f9=-1:h=ABS(h):divh=1:t=u0
1710 IF f9rac<0 GOTO 1640 ELSE 2200
2000 '======================================================== Raccordement
2010 IF j=0 GOTO 3000
2020 j=j-1
2030 x=rac(j,0):y=rac(j,1):z=rac(j,2)
```

```
2040 IF rac(j,3)=10 GOTO 2000 ELSE h=h*rac(j,3):t=rac(j,4)
2050 GOSUB 1400:PLOT XX,YY:GOSUB 1300
2060 FOR w%=1 TO 3:GOSUB 1400:DRAW XX,YY:GOSUB 1300:NEXT
2070 FOR u=0 TO n:prod(u)=sign(u)*(pl(u,0)+pl(u,1)*x+pl(u,2)*y+pl(u,3)*z):
NEXT
2080 GOTO 1100
2100 '========================================= Test d'egalite des raccords
2110 FOR w=0 TO j-1
2120 IF (x-rac(w,0))^2+(y-rac(w,1))^2+(z-rac(w,2))^2<h*h*4 THEN fgrac=w:
RETURN
2130 NEXT
2140 fgrac=-1:RETURN
2200 '============================================ Oter un point de raccord
2210 FOR w=fgrac TO j-2
2220 FOR ww=0 TO 4:rac(w,ww)=rac(w+1,ww):NEXT
2230 NEXT:j=j-1
2240 GOTO 2000
2300 '========================================= Calcul du Polynome au Point
2310 p=p(0)+p(1)*x+p(2)*y+p(3)*z+p(4)*x*x+p(5)*y*y+p(6)*z*z+p(7)*x*y+
p(8)*x*z+p(9)*y*z
2320 RETURN
2500 '===================================================== Test Polynomial
2510 GOSUB 2300
2520 IF ABS(p)<0.05 THEN RETURN
2530 x=x0:y=y0:z=z0
2600 '========================================================== Correction
2610 GOSUB 2300:p0=p
2620 g=-0.1*SGN(p)
2630 NN0=p(1)+2*x*p(4)+y*p(7)+z*p(8)
2640 NN1=p(2)+2*y*p(5)+x*p(7)+z*p(9)
2650 NN2=p(3)+2*z*p(6)+x*p(8)+y*p(9)
2660 x=x+g*NN0:y=y+g*NN1:z=z+g*NN2:GOSUB 2300
2670 IF ABS(p)<0.005 THEN RETURN
2680 IF p*p0<0 THEN g=-g/2
2690 p0=p:GOTO 2660
3000 '================================================================ FIN
3010 PRINT CHR$(7)
3020 WHILE INKEY(47)=-1 AND INKEY(79)=-1 AND INKEY(53)=-1:WEND
3030 IF INKEY(47)<>-1 THEN MODE 1:INK 1,24:PEN 1:GOTO 700
3040 IF INKEY(79)<>-1 THEN RUN
3050 IF INKEY(53)<>-1 THEN MODE 1:INK 1,24:PEN 1:INK 0,1:BORDER 1:END
```

Évidemment, plus le degré est élevé, plus le nombre de monômes possibles est important. Ainsi, pour le degré 0, un seul mônome est possible : $u = v = w = 0$, donc pas de x, ni de y, ni de z ; la constante a est toute seule, c'est le *terme constant*. Pour le degré 1, seuls trois monômes sont possibles : $a \cdot x$; $a \cdot y$; $a \cdot z$. Un polynôme de degré 1 est du type : $a \cdot x + b \cdot y + c \cdot z + k = 0$ (a, b, c, k, quatre constantes) ; c'est donc un plan, la surface la plus simple. Enfin, il peut y avoir six termes de degré 2 (voir lignes 100 à 150 du programme) : un polynôme de degré 2, ayant des monômes de degré 0, 1 ou 2, a donc dix monômes différents, qui sont tous précédés d'une constante. Dans le programme, ce sont ces dix constantes qui sont demandées en lignes 50 à 160, et stockées dans une petite matrice p. Par exemple, si vous voulez rentrer l'équation d'une sphère de rayon 1, soit

$$x^2 + y^2 + z^2 - 1 = 0$$

vous devrez rentrer dans l'ordre : $-1, 0, 0, 0, 1, 1, 1, 0, 0, 0$.

Ensuite, on demande les plans limites. Ce sont des plans où la surface sera arrêtée. Le programme tracera aussi l'intersection de ces plans avec la surface, dans leur partie visible (s'il y en a une). Ces plans sont nécessaires en général, car les surfaces polynomiales sont presque toujours infinies, au moins dans une direction. C'est dans cette (ou ces) direction(s) qu'il faudra mettre un plan qui arrêtera la surface, sinon le tracé va sortir de l'écran. Le programme (lignes 170 à 250), demande de rentrer l'équation du plan sous la forme $u \cdot x + v \cdot y + w \cdot z + k$ (u, v, w, k étant quatre constantes quelconques). Toutefois, pour des raisons que nous expliquerons plus loin, vous ne devez pas entrer 0 pour w, sinon un message d'erreur se produirait.

La précision du tracé est demandée ensuite. En effet, le tracé des courbes se fait d'une manière approximative, en traçant de petits bouts de droite successifs ; h représente la longueur de ces petits bouts. La courbe aura bien l'air d'en être une si h est très petit $(0,01)$, mais le tracé sera plus long. Cela est complètement général, quelle que soit la courbe tracée, plane ou de l'espace, etc. (voir Annexe A). Ici, $h = 0,05$ convient en général.

Toujours expliquant le programme, je passe pour l'instant sur les coefficients différentiels pour arriver au calcul du point initial. En effet, le tracé de toutes ces courbes (horizon + intersections avec les plans limites) se fait point par point (comme pour toute courbe), c'est-à-dire que chaque point est déduit du précédent, comme nous le verrons dans un moment. Il faut un point de départ (éternel problème de l'algorithme et de l'amorce). Ce point de départ sera lui-même déduit d'un point initial, calculé une bonne fois pour toutes (lignes 500 à 600). Ce point est calculé sur le plan limite 1 (qui existe nécessairement de ce fait). Si ce plan a pour équation $z = \lambda \cdot x + \mu \cdot y + v$, en remplaçant z par sa valeur (fonction de x et de y) dans l'équation de la surface, on obtient une équation du second degré en x et y, du type :

$$u \cdot x^2 + v \cdot y^2 + w \cdot x \cdot y + a \cdot x + b \cdot y + c = 0$$

où u, v, w, a, b, c sont six constantes calculées par le programme (ligne 510). Je ne m'étendrai pas sur la résolution de cette équation, qui aboutit à des équations du second degré et d'une variable, du type :

$$aa \cdot y^2 + bb \cdot y + cc = 0$$

dont la résolution est aisée. Mais songez que si la surface était de degré supérieur, l'équation ici le serait aussi, et le calcul du point initial serait assez compliqué (résolution pratique nécessaire : voir comment résoudre une équation de manière approchée plus loin). C'est une des raisons pour lesquelles je me suis limité au degré 2. La seconde ne va pas tarder à apparaître. Je dois pour cela expliquer comment l'on peut tracer l'horizon d'une surface polynomiale.

En un point donné d'une surface, mettons M, de coordonnées (x, y, z) (vérifiant l'équation de la surface, évidemment), on peut presque toujours définir un vecteur normal à la surface. C'est le vecteur normal au plan tangent à la surface. Pour vous représenter ce plan, imaginez que vous ayez fabriqué un modèle en bois de la surface. Sur ce modèle, vous faites une croix au crayon au point M. Vous prenez ensuite une petite planche carrée (et plane), et vous la clouez sur la surface, exactement au point M. Eh bien, le plan de cette planche, qui collera au plus près à la surface, est confondu avec ce que l'on appelle le plan tangent à la surface. Sa perpendiculaire (le clou), porte le vecteur dit normal à la surface, qui est dirigé dans le sens inverse du clou, de façon à avoir l'air de sortir de la surface (comme pour les polyèdres). Tout cela est résumé par le schéma *a* de la Figure VII.D. Voyez aussi l'Annexe A à ce sujet.

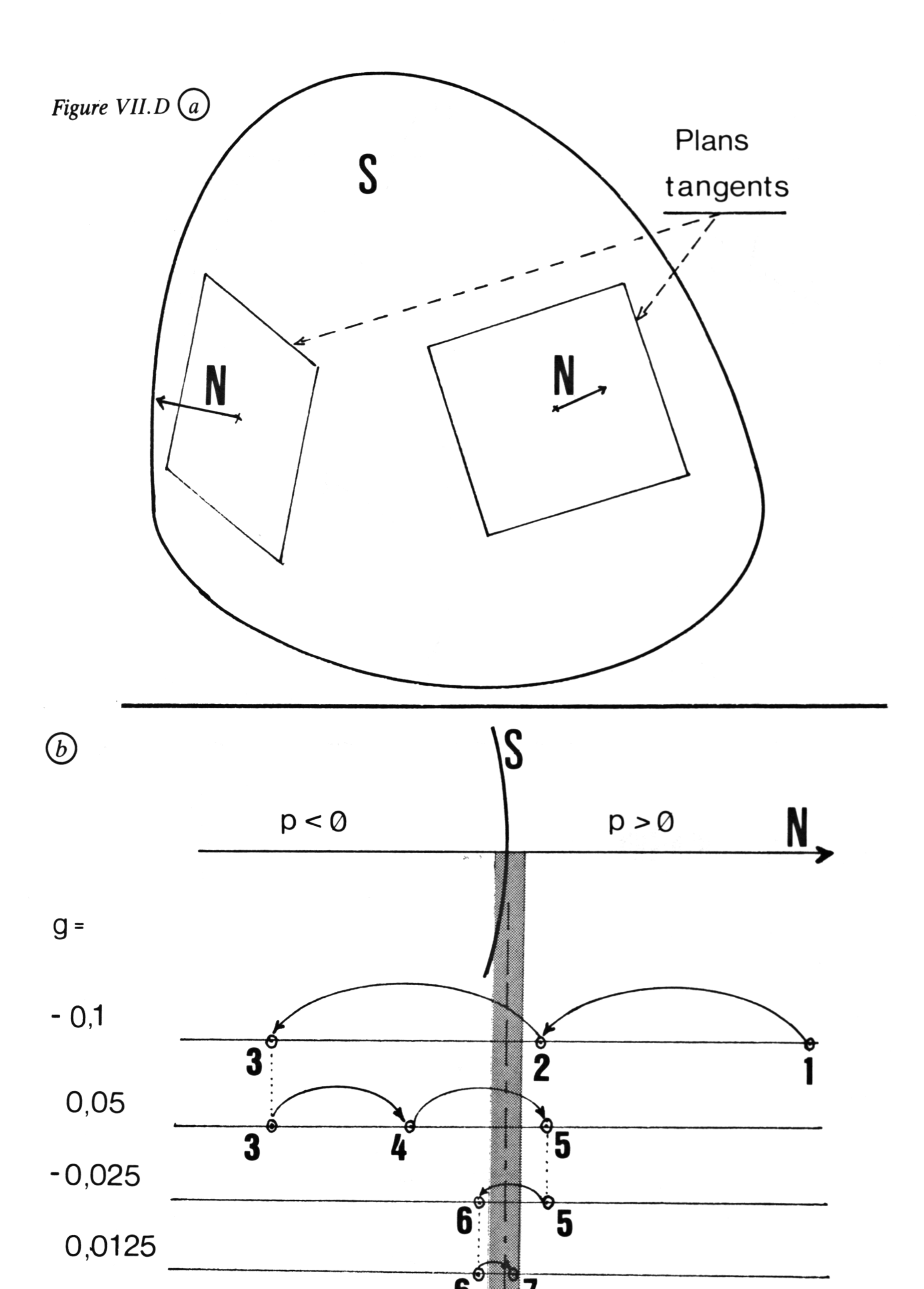


Figure VII.D (a)
S
Plans tangents
N
N
(b)
S
p < 0
p > 0
N
g =
- 0,1
3
2
1
0,05
3
4
5
-0,025
6
5
0,0125
6
7

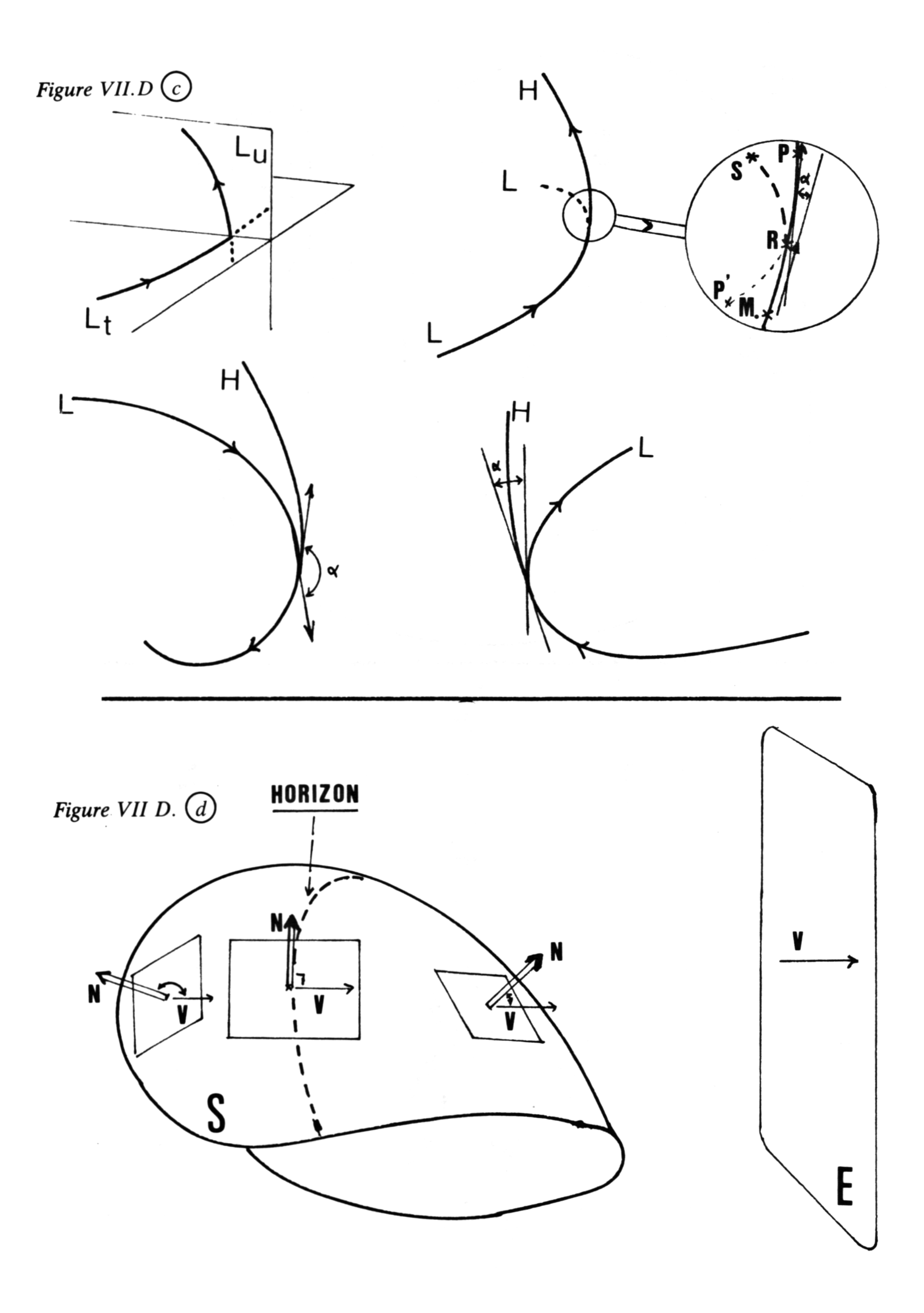


Figure VII.D (c)
Lu
Lt
H
L
L
S
P
α
R
P'
M.
H
L
α
H
L
α
Figure VII D. (d)
HORIZON
N
N
N
V
V
V
S
V
E

Le vecteur normal en un point est calculable. Ses coordonnées (p, q, r), sont les dérivées partielles de l'équation de la surface, respectivement par rapport à x, à y et à z (pour plus de précisions sur ce point, voir l'Annexe A sur la dérivation).

D'autre part, nous connaissons les coordonnées (v_0, v_1, v_2) de la normale **V** à l'écran. Ces dernières, à la différence de p, q et r, sont immuables car elles ne dépendent que des deux angles psi et thêta du point de vue.

Un point est sur l'horizon si la normale en ce point est perpendiculaire à la normale à l'écran, c'est-à-dire si leur point scalaire est nul. L'horizon H a donc pour équation :

$$v_0 \cdot p + v_1 \cdot q + v_2 \cdot r = 0$$

où p, q, et r sont en fait des fonctions de x, y, z. Ces fonctions sont aussi des polynômes, de degré égal au degré de la surface moins un. C'est pourquoi, pour des surfaces de degré 2, cette équation de l'horizon est de degré 1 ; c'est donc l'équation d'un *plan*. C'est là la chose fondamentale, propriété essentielle du degré 2 : l'horizon est sur un plan, comme pour les plans limites ; ce sera le plan limite zéro, dont les coefficients sont calculés aux lignes 770 et suivantes.

Si la surface était de degré supérieur, l'horizon ne serait pas contenu dans un plan, et le tracé en serait un peu plus complexe ; c'est la raison pour laquelle je me suis limité au degré 2 dans ce programme quoique, en réalité, ce ne soit pas obligatoire (voyez l'Annexe B).

Encore un mot sur cette définition de l'horizon par un produit scalaire. Elle ne doit pas vous surprendre. En effet, souvenez-vous que pour les polyèdres, lorsque le même produit scalaire était nul, la face était vue par la tranche ; elle était alors à la limite du polyèdre. Ici, c'est le plan tangent qui est vu par la tranche, car le point M est à la limite de la surface, sur l'horizon (voir Figure VII.D, schéma *b*). C'est pour la même raison que le test de visibilité (lignes 1200 et suivantes) consiste à chercher le signe de ce produit scalaire. La différence principale réside dans le fait que ce produit scalaire change à chaque point, alors que pour les polyèdres il changeait à chaque face. Mais en fait, une surface bombée est une sorte de polyèdre où chaque point est une minuscule face.

Précisons toutefois, une fois encore, que l'horizon pourrait très bien être tracé, même pour des polynômes d'ordre supérieur. Expliquons tout de suite comment votre ordinateur va pouvoir tracer l'intersection de la

surface d'équation $P(x, y, z) = 0$ avec un plan (plan limite) d'équation $a \cdot x + b \cdot y + c \cdot z + k = 0$.

Tout d'abord, cette dernière équation doit être convertie, sous la forme $x = \ldots$, $y = \ldots$ ou $z = \ldots$, suivant que a, b ou c est non nul. Ici nous supposerons que c n'est pas nul. Alors on peut écrire :

$$z = \frac{(a \cdot x + b \cdot y + k)}{(-c)}$$

Reste alors à remplacer z par sa valeur dans l'équation $P(x, y, z) = 0$, pour obtenir une équation du type $Q(x, y) = 0$, où Q est aussi un polynôme de degré 2. Par exemple, si $P(x, yz) = x \cdot y + 4 \cdot y + 6 + y \cdot z + 7 \cdot z$, et que le plan limite a pour équation $-x + z - 3 = 0$, c'est-à-dire $z = 3 + x$, on remplace z par $3 + x$ et l'on obtient :

$$Q(x, y) = x \cdot y + 4 \cdot y + 6 + y \cdot (3 + x) + 7 \cdot (3 + x)$$

$$= x \cdot y + 4 \cdot y + 6 + 3 \cdot y + x \cdot y + 21 + 7 \cdot x$$

soit :

$$Q(x, y) = 2 \cdot x \cdot y + 7 \cdot x + 7 \cdot y + 27$$

qui est un polynôme en x et y d'ordre 2 aussi. D'une façon générale, un tel polynôme a six termes : constant, en x, en y, en x^2, en y^2, en $x \cdot y$. Ils sont ici égaux à $27, 7, 7, 0, 0, 2$. Pour chaque plan limite, le programme calcule ces termes aux lignes 300 et suivantes. Y compris pour l'horizon, ici considéré comme le plan limite zéro (ce ne serait pas possible pour un degré plus important, qui exige une double résolution d'équation implicite).

Cette équation $Q(x, y) = 0$ est appelée équation implicite en x et y. Pour le détail de sa résolution point par point, le lecteur se reportera à l'Annexe B. Cette résolution est ici exécutée par le programme aux lignes 1300 et suivantes : à partir d'un point, sur l'un des plans limites, le programme peut trouver le point suivant sur la courbe, située à la distance h. Les courbes, intersections des plans limites et de la surface peuvent donc ainsi être tracées.

C'est ce que fait le programme. Partant du point initial, il cherche, sur la courbe C_1, intersection du plan limite 1 (Pl_1) avec la surface S, le premier

point qu'il trouvera visible (test de visibilité en 1200, d'après le fameux produit scalaire). Ce point, qui est souvent le point initial lui-même, est appelé point de départ. En effet, c'est à partir de lui que tout commence. On le place sur l'écran, après avoir calculé ses coordonnées projetées XX et YY (toujours les mêmes formules...). Puis on cherche le point suivant, on le place, et on recommence.

Deux cas peuvent faire changer cette routine. Tout d'abord le retour au point de départ (obligatoire en principe, sinon c'est qu'il manque un plan limite). On passe alors à d'autres tracés, s'il y a lieu, comme nous le verrons. Second cas, très important celui-là, le changement de plan limite.

En effet, les courbes $C_0, C_1, C_2, \ldots$, intersections des plans limites avec la surface, se rencontrent en général entre elles et c'est heureux, sans quoi il serait difficile d'aller les tracer. Il faudrait en effet recommencer le même travail que pour le plan 1 (calcul du point initial), travail déjà pas simple, mais absolument abominable pour des degrés supérieurs. Comment savoir si le point où l'on se trouve n'est pas tout proche d'un autre plan limite ? Tout simple, il suffit d'appliquer l'équation de ce plan au point, et l'on obtient la valeur s (ligne 1460). Si s est nul, le point est sur le plan, mais cela n'arrive jamais ; le point est en fait, au mieux, tout près du plan ; s est alors très faible et cela suffit (les tracés de courbes sont de toute façon toujours très approximatifs).

Une précision sur le signe de s : selon qu'il est du même signe que le s du point de départ (signe stocké, pour chaque plan limite, dans la matrice sign) ou non, le point considéré est du même côté que le plan (alors prod est positif), ou du côté opposé (alors prod est négatif).

Conclusion : lorsque prod change de signe, ce qui se traduit par prod · prod (u) négatif, c'est qu'on a traversé le plan limite n° u. Dès lors, il ne faut pas tracer, mais rechercher le point intermédiaire, qui est à la fois sur la courbe t (que l'on était en train de tracer) et sur la courbe u. Ce point est le point de raccord entre les deux courbes. On en trouve la valeur précise en oscillant autour, en diminuant h pour faire des pas de plus en plus petits. Le point est enfin (pratiquement) trouvé lorsque prod est petit (ligne 1480). On part alors vers une routine spéciale qui traite les points de raccord.

Il existe deux types principaux de points de raccord : les raccords immédiats, et les raccords reportés. Dans le cas des premiers (voir Figure VII.D, schéma *c* 1), deux courbes aboutissent au point de raccord, mais n'en repartent pas, en ce sens que leurs prolongements au-delà de ce point sont soit cachés (car les parties cachées sont pour l'essentiel non

représentées), soit au-delà d'un plan limite donné. Or, l'on ne représente que la partie de la surface qui se trouve du même côté que le point de départ par rapport à chacun des plans limites. Lorsqu'une courbe C_t traverse le plan L_u, le point d'intersection, étant à la fois sur C_t, donc sur la surface, et sur L_u, se trouve sur C_u : c'est bien un point de raccord entre les deux courbes C_t et C_u, et l'on passe alors de l'une à l'autre dans le tracé, d'où le nom de *changement de plan limite* qui a été donné à cette routine.

Malheureusement, ce nom devient injustifié lorsqu'on se trouve dans le cas d'un raccord reporté.

Ce genre de cas peut se produire lors de l'intersection de l'horizon avec une courbe d'un plan limite L, ou (dans des cas exceptionnels) entre deux plans limites.

C'est surtout avec l'horizon H qu'il y a des problèmes. Examinons en effet la Figure VII.D, schéma *c*, n°s 2 à 4. En 2, que voyons-nous ? Le tracé arrive par L (suivre les flèches), puis rencontre l'horizon H ; la courbe de L disparaît derrière la surface après cette rencontre. Le tracé doit donc se poursuivre sur H, en abandonnant L. En 3, le tracé arrive sur L, rencontre l'horizon, mais doit se poursuivre car L n'est pas dissimulée derrière la surface au-delà. Il en est de même en 4.

C'est pourquoi le programme doit procéder à des tests. Examinez la partie agrandie de 2 (dans le grand cercle). Arrivant de M_0 (coordonnées projetées XX0, YY0), la courbe arrive en R, point de raccord (après une recherche dont nous avons déjà parlé, et dont nous reparlerons). Sur L toujours, après R (coordonnées projetées XX*c*, YY*c*), il y a S (calculé en ligne 1540 dans le programme). Pour S, le *flag indic* montre si ce point est visible (c'est-à-dire sur la partie avant de la surface) comme dans le schéma *c*4, ou invisible comme dans les schémas *c*2 et *c*3. Observez bien en effet, dans ce dernier cas, que la partie de L située au-delà du point de raccord est visible en pratique (c'est-à-dire qu'il faut la tracer), mais invisible en théorie, car derrière la surface ; la différence vient de ce que toute la partie de la surface qui masquait ce second morceau de courbe se trouvait en dessous du plan de L, et par conséquent a été en quelque sorte arrachée, enlevée.

Mais R appartient aussi à H. Sur H, il y a deux points suivant R : P et P′, suivant la direction dans laquelle on va. Des deux, il faut choisir celui qui se trouve du bon côté du plan de L (au-dessus dans les cas de figure), soit P ; cela est fait aux lignes 1550 à 1570.

Ensuite le programme examine l'angle séparant MR de RP. Si cet angle est aigu (cas 2 et 4), le produit scalaire *ps* de ces deux vecteurs est positif, sinon (cas 3) il est négatif.

Il suffit de regarder les cas de figure pour constater qu'il faut à la fois que S soit invisible (cas 2 et 3) et que *ps* soit positif (cas 2 et 4) pour que l'on puisse effectivement passer d'une courbe à l'autre (cas 2). Dans les autres cas (3 et 4), le point de raccord R est enregistré dans une matrice spéciale *rac :* lorsque le tracé sera fini (retour au point de départ), la routine *raccordement* se chargera de faire repartir le tracé de ce point de raccord.

Précisons d'autre part que, de toute façon, R sera enregistré (mais avec fg = 10 au lieu de fg = ± 1), afin que si le tracé, par hasard, venait à revenir en ce point, il soit stoppé (car il est douteux que quatre courbes se rejoignent au même point ; le tracé se recopierait donc, sans cette précaution).

Voilà l'essentiel de ce qu'il fallait dire au sujet de cette routine *changement de plan limite*, et de ses auxiliaires, les trois suivantes. Notez simplement que *j* est le pointeur de rac, qui sert de pile ici.

Cependant, vous n'avez peut-être pas très bien compris comment l'on trouve une valeur assez exacte du point de raccord. Si c'est le cas, je vous propose de le développer plus en détail, en expliquant une autre routine effectuée pour chaque point en ligne 2500. Cette routine a en effet valeur d'exemple puisqu'elle est le modèle classique pour trouver la solution d'une équation.

Pour un point donné (x, y, z), la valeur $p = P(x, y, z)$ est calculée par la routine en ligne 2300. Idéalement, puisqu'on trace la surface d'équation $P(x, y, z) = 0$, on devrait avoir constamment $p = 0$.

Malheureusement ce n'est pas le cas, et rien ne peut y faire. En effet, on ne peut tracer ce genre de courbe ou de surface qu'approximativement. Conséquence, p n'est pas nul. Pire même : au fur et à mesure que l'on trace, les petites erreurs de calcul qui se produisent à chaque fois s'accumulent, tant et si bien qu'on finit par dépasser les bornes du tolérable ; p devient, en valeur absolue, supérieur à une valeur déjà plus tout à fait très petite : 0,05 (ligne 2520). Que faire alors ? Une correction (lignes 2600 et suivantes).

Examinons le schéma *d* (toujours de la Figure VII.D). En haut, la surface est vue par la tranche. Elle est accompagnée de sa normale N. Cette normale a été représentée encore quatre fois en dessous, pour ne pas surcharger la figure.

Lorsque cette routine est appelée, (x, y, z) sont les coordonnées du point 1. Ce point étant jugé trop éloigné de la surface, on va tâcher de le recentrer en le replaçant tout près de la surface, dans la zone grisée où p est, en valeur absolue, inférieur à 0,005.

Pour cela, on va se déplacer sur la normale, à une distance g. Notez que la surface partage l'espace en deux parties : d'un côté (ici à droite) p est positif, de l'autre il est négatif, et nul sur S, bien entendu. Il faut rendre p, en valeur absolue, le plus petit possible. Ayant calculé p au point 1, si p est positif, il faut le diminuer (d'où g négatif), et sinon l'augmenter (g positif), afin, dans tous les cas, de le rapprocher de la valeur parfaite : zéro. Donc, en ligne 2620, g est initialisé avec le signe inverse de p, à $-0,1$ dans le cas de figure (g est indiqué à gauche de la figure).

On calcule alors le point suivant sur la normale, et l'on arrive au point 2 (ligne 2660). On voit que 2 n'est pas dans la zone grise (ligne 2670), il faut donc continuer. D'autre part, p, en 2, est du même signe que p, en 1. Il faut continuer à le diminuer. On arrive ainsi en 3, sans que g ait changé.

Arrivé en 3, p est devenu négatif. Il faut donc l'augmenter, donc changer le signe de g. Mais ce n'est pas tout. Si l'on ne fait que cela, on va revenir en 2, puis en 3, puis en 2,...

Le truc consiste alors à diviser g par deux. g vaut alors $0,05$ (ligne 2680). On arrive en 4, p reste négatif. On passe en 5 où p redevient positif (5 est le même point que 2). On repart dans le sens inverse, non sans avoir divisé g par deux et avoir changé son signe. On arrive alors en 6, et remarquez que l'on se rapproche de plus en plus du centre. En 6, p devient négatif. On redivise g par deux, on change son signe, et l'on arrive en 7 qui se trouve dans la zone grise : c'était le résultat recherché. Il n'y a plus qu'à repartir de 7 pour continuer le tracé. C.Q.F.D.

Sur le dessin, la différence entre le point 1 et le point 7 est infime, mais elle est essentielle, sinon le programme se perdrait rapidement.

Notez toutefois que toute cette correction ne s'effectue pas tout à fait sur la même droite, car les coordonnées du vecteur normal sont recalculées en chaque point pour aboutir plus rapidement.

Retenez ce raisonnement, car c'est *le* raisonnement de base pour trouver, de manière approchée, la solution d'une équation. Il en existe beaucoup d'autres, plus ou moins subtils, mais celui-là est indispensable. Voyez à ce sujet l'Annexe B.

Il ne vous reste plus guère qu'à tester le programme avec les exemples qui vont suivre. Je dois simplement vous préciser deux points de détail. Tout d'abord l'échelle est un coefficient multiplicatif qui permet de rendre

plus ou moins grand le dessin. Si vous ne voyez qu'un petit tas, agrandissez sa valeur (vers 50), sinon diminuez-la. D'autre part, lorsque le tracé est fini, vous entendez un bip sonore. Alors, si vous appuyez sur *f* (comme fin), tout s'arrête (ligne 3050). Si vous appuyez sur DEL, tout recommence au début. Si vous pressez la barre, vous retournez en 700 choisir de nouveaux angles pour changer le point de vue.

Toutefois, avant de donner les exemples les plus significatifs, je dois vous mettre en garde. Le programme ne marchera pas dans tous les cas, tant s'en faut. Il lui arrivera assez fréquemment de se “planter”, d'une manière ou d'une autre.

Les deux plus courantes des erreurs seront celles-ci. Soit le tracé s'arrête et ne reprend pas. C'est que le programme boucle entre la correction et le passage au point suivant. Soit le tracé passe à côté du point de départ sans s'arrêter : l'erreur, au niveau de l'équation des plans, est devenue trop importante. Dans un cas comme dans l'autre, la solution la plus efficace est de diminuer la précision du tracé ; recommencez en la divisant par deux. Dans le second cas, toutefois, un simple changement des angles peut suffire à régler le problème, et c'est tant mieux car diminuer *h* revient à augmenter la durée du tracé.

D'autre part, il peut arriver des problèmes par votre faute : le choix des plans limites n'est pas toujours facile. En effet, il est nécessaire que le premier rencontre la surface ; il faut donc en être sûr. Même si c'est le cas, le tracé peut sortir de l'écran et ne pas revenir ; c'est qu'il manque un plan.

Enfin, un problème qui n'a pas de réelle solution : les points anguleux. Ce sont les points (heureusement très rares) où il n'existe pas de plan tangent. C'est le cas notamment de la pointe des cônes. Dès lors, il peut arriver (pas toujours) des ennuis au moment où le tracé arrive en de tels points. N'oubliez pas, en particulier, qu'il n'y a pas de vecteur normal à cet endroit, d'où problème... Là encore, la diminution de *h* peut être le remède salvateur.

Et puis, il peut arriver quelque chose que je n'ai pas prévu. Cela ne m'est pas encore arrivé, depuis que ce programme est au point (dans la limite de ses possibilités). Dans ce cas, essayez encore la même méthode si cela se produit, ce que je n'espère pas. Mais, en tout état de cause, pas d'illusions, ce programme n'a rien de miraculeux. Par contre, il peut tracer sans aucun problème des morceaux pas trop grands ni trop difformes de surfaces bombées, or c'est cela sa véritable utilité, si vous vous souvenez de ce que j'ai dit sur la C.A.O.

4. EXEMPLES

Voici donc les types principaux de surfaces que vous pourrez représenter. Ils sont tous suivis d'un exemple concret, donné par des chiffres qui sont les données que vous devez rentrer, *dans l'ordre*, dans le programme. La barre indique les moments où l'écran s'efface.

Tout d'abord la sphère :

$$x^2 + y^2 + z^2 - R^2 = 0$$

(R est le rayon, une constante). Pas besoin d'épiloguer.

Exemple :

$$-4, 0, 0, 0, 1, 1, 1, 0, 0, 0 / 1, \quad 0, 0, 1, 0 / 0.1 / 100, 45, 45$$

qui vous donnera une demi-sphère.

Variation sur la sphère, l'ellipsoïde est une sorte de ballon de rugby :

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - k^2 = 0$$

(a, b, c, k, quatre constantes non nulles).

Exemple :

$$-4, 0, 0, 0, 1, 1, 2, 0, 0, 0 / 1, \quad 0, 0, 1, 0 / 0.1 / 80 / 45, 45$$

L'hyperboloïde à une nappe rappelle les tours de refroidissement des centrales :

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} - k^2 = 0$$

(idem pour a, b, c, k).

Exemple :

$$-1, 0, 0, 0, 1, 1, -1, 0, 0, 0 / 2, \quad 0, 0, 1, 1, \quad 0, 0, 1, \quad -1 /$$

$$0.05 / 100 / 60, 75$$

L'hyperboloïde à deux nappes est théoriquement en deux parties, mais vous ne pouvez en voir qu'une car le programme ne peut tracer deux morceaux disjoints. Il faudrait donc le faire repartir, par exemple, pour en voir plus (ce n'est pas passionnant, de toute façon) :

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}-\frac{z^2}{c^2}+k^2=0$$

Exemple :

+ 1, 0, 0, 0, 1, 1, – 1, 0, 0, 0 / 1,   0, 0, 1, 2 / 0. 05 / 50 / 45, 60

Le paraboloïde ressemble au précédent, mais en plus rondouillard :

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}-z=0$$

Exemple :

0, 0, 0, 1, 1, 1, 0, 0, 0, 0 / 1,   0, 0, 1, 1 / 0. 05 / 80 / 45, 45

Le cône, qui tient le milieu entre les deux hyperboloïdes, est une forme bien connue qui provient du conoïde :

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}-z^2=0$$

dans le cas où $a=b$. Il est très difficile à représenter par ce programme, en grande partie à cause de son sommet qui est un point anguleux. Ce point sera donc généralement mal représenté, avec une sorte de brouillage (voir figure VII.C.).

Exemple :

0, 0, 0, 0, 1, 1, – 1, 0, 0, 0 / 2,   0, 0, 1, 1, 0, 0, 1, – 1 / 0. 05 / 80 / 45, 60

Vous pouvez voir ces exemples sur les Figures VII.C, nos 1 à 6. Il va sans dire que vous pouvez utiliser bien d'autres types d'équations. Mais vous retomberez presque toujours sur un de ces types ; simplement, la figure sera orientée différemment, ou plus grosse. Rien ne vous empêche de faire des essais au hasard, surtout si vous aimez les surprises...

Si vous préférez passer à des choses plus simples (comme je vous comprends !), changez simplement de chapitre, en tournant la page...

VIII

L'OMBRE ET LE FOND DES CHOSES : COUPONS, TRANCHONS, PLAQUONS

Après avoir vu comment créer et tracer simplement certains objets, et avant de nous attaquer à une conception plus exacte de la vision, au Chapitre X, nous allons citer ici quelques petits trucs pas trop difficiles qui permettent de rendre vraiment plus réalistes les travaux de votre ordinateur et également de les analyser plus en profondeur.

Si vous savez à présent comment tracer des objets complexes, vous ne savez peut-être pas encore comment en tracer l'ombre, par exemple, ou le reflet dans un miroir, une vitre, ou une étendue d'eau. Or, ce sont précisément ces détails qui rendent vraiment réalistes les dessins en trois dimensions.

D'autre part, les objets un peu compliqués sont toujours plus faciles à comprendre si on les décortique. Votre ordinateur vous permet de le faire sans "casse" ; il serait dommage de ne pas en profiter. Ainsi, pour des utilisations presque professionnelles aussi bien que dans les jeux, les coupes et sections se révèlent parfois d'une étonnante utilité. C'est ce que nous verrons dans la seconde partie de ce chapitre.

1. OMBRES ET REFLETS : LES MIRACLES DE LA PROJECTION

Nous sommes à présent bien familiers avec la projection car, en fait, c'est ce que nous ne cessons de faire depuis le début. Toutefois nous ne connaissons qu'un type de projection, la projection orthogonale. Rappelons que ce qualificatif provient de ce que le segment joignant un point et son projeté est toujours perpendiculaire au plan de projection, l'écran en l'occurrence. Pour nous, cette définition rébarbative se traduisait par une réalité bien simple : dans le repère lié à l'écran, où son équation est Z = constante, il suffisait de ne pas tenir compte de la troisième coordonnée ZZ ; il en résultait qu'il suffisait de calculer XX et YY, ce qui était bien commode.

PROGRAMME VIII.1

```
530 e=150/SQR(rmax)
550 WINDOW #1,21,40,1,25
560 alpha=45:fga=1
605 fgM=0
615 n0=s1*s4+c4*c1*s2:n1=-s4*c1+c4*s2*s1:n2=c2*c4
625 IF fgM=0 THEN Prod=c1*s2*n(t,0)+s1*s2*n(t,1)+c2*n(t,2) ELSE
Prod=n0*n(t,0)+n1*n(t,1)+n2*n(t,2)
730 IF fgM=1 GOTO 2000
740 IF fga=1 GOTO 4050
750 fgM=1
760 GOTO 620
1100 PLOT -160,0,3
1120 DRAW 150*s1-160,-150*c1*c2:PRINT"x";
1130 PLOT -160,0
1140 DRAW -150*c1-160,-150*s1*c2:PRINT"y";
1150 PLOT -160,0
1160 DRAW -160,150*s2:PRINT"z";
1215 XX=XX-160
1230 IF fgM=0 THEN RETURN
1240 ZZ=(s2*c1*x+s2*s1*y+c2*z)*e
1250 XX=-(XX+160)*c4+ZZ*s4+160
1260 RETURN
2070 a=INKEY(54):IF a<>-1 THEN GOSUB 1900:alpha=alpha-b:GOTO 4000
2080 a=INKEY(46):IF a<>-1 THEN GOSUB 1900:alpha=alpha+b:GOTO 4000
2090 GOTO 2010
2645 PRINT:PRINT"_ B ou N pour faire tourner le miroir."
2825 PRINT:PRINT"_ B ou N pour faire tourner le miroir."
4000 '============================================= Rotation du miroir
4010 s3=SIN(alpha):c3=COS(alpha)
4020 s4=SIN(2*alpha):c4=COS(2*alpha)
4030 CLS #1
4040 IF s3>0 AND c3>0 THEN fga=0:fgM=1:GOTO 615
4050 PEN 2:LOCATE 21,10
4060 PRINT"Miroir mal oriente"
4070 PEN 1:GOTO 2000
```

Cela ne le sera plus, et dans le Programme VIII.1 il faudra calculer ZZ (ligne 1240). Avant d'expliquer pourquoi et ce que fait ce programme, je dois préciser qu'il est destiné à être amalgamé avec les Programmes VI.2 (et ceux qui l'accompagnent) et VI.5, qui forment un tout permettant de tracer tous les polyèdres de répétition cylindrique, avec parties cachées. Il en sera de même des deux autres programmes de ce chapitre. J'ai choisi ce type de polyèdres car ils sont les plus simples, mais évidemment, rien ne vous empêche de réadapter les programmes de ce chapitre pour tous les objets que nous sommes parvenus à tracer.

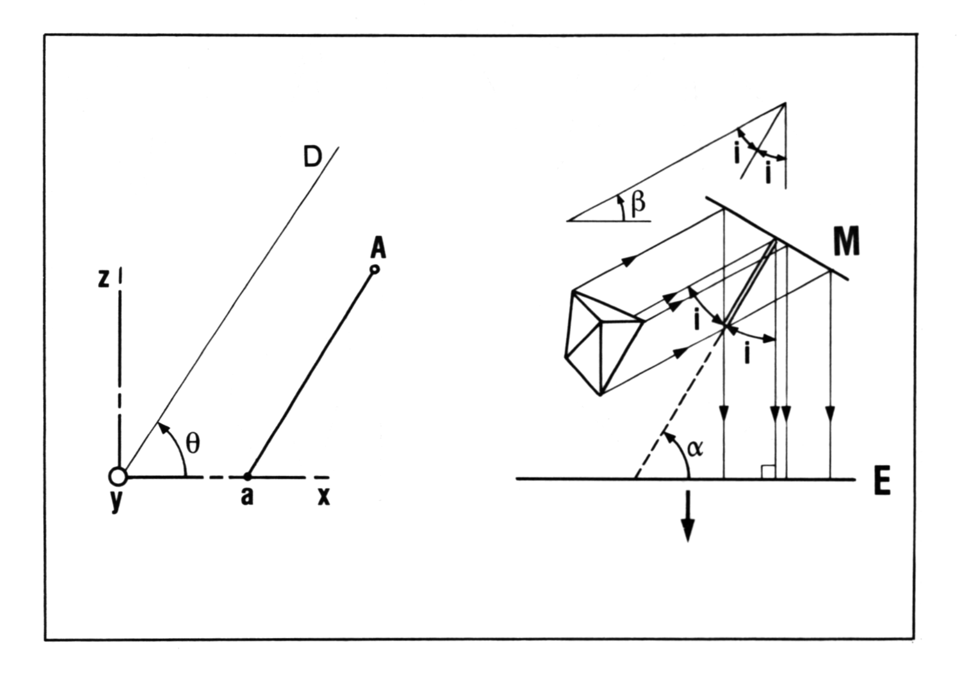


Figure VIII.1 (a)

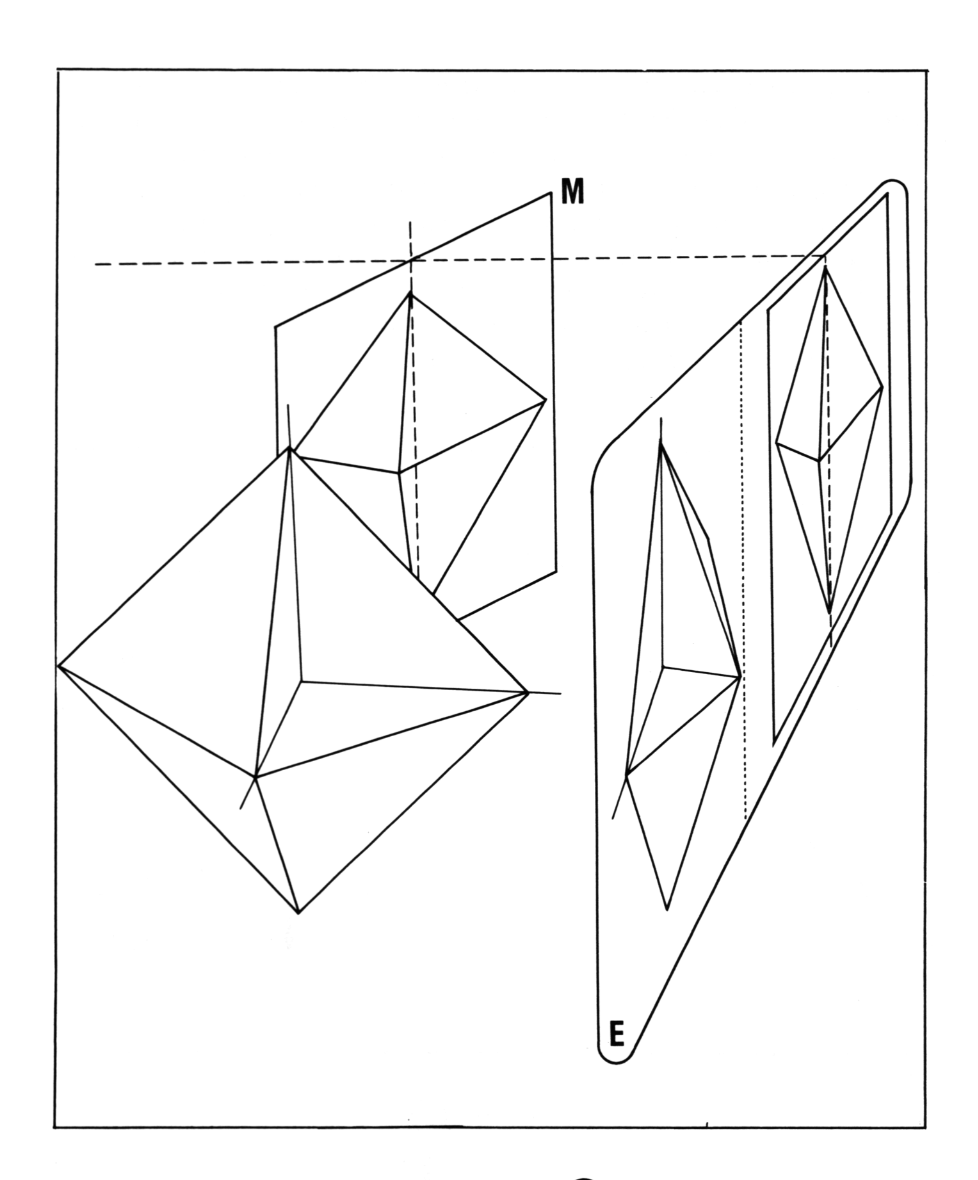


Figure VIII.A (b)

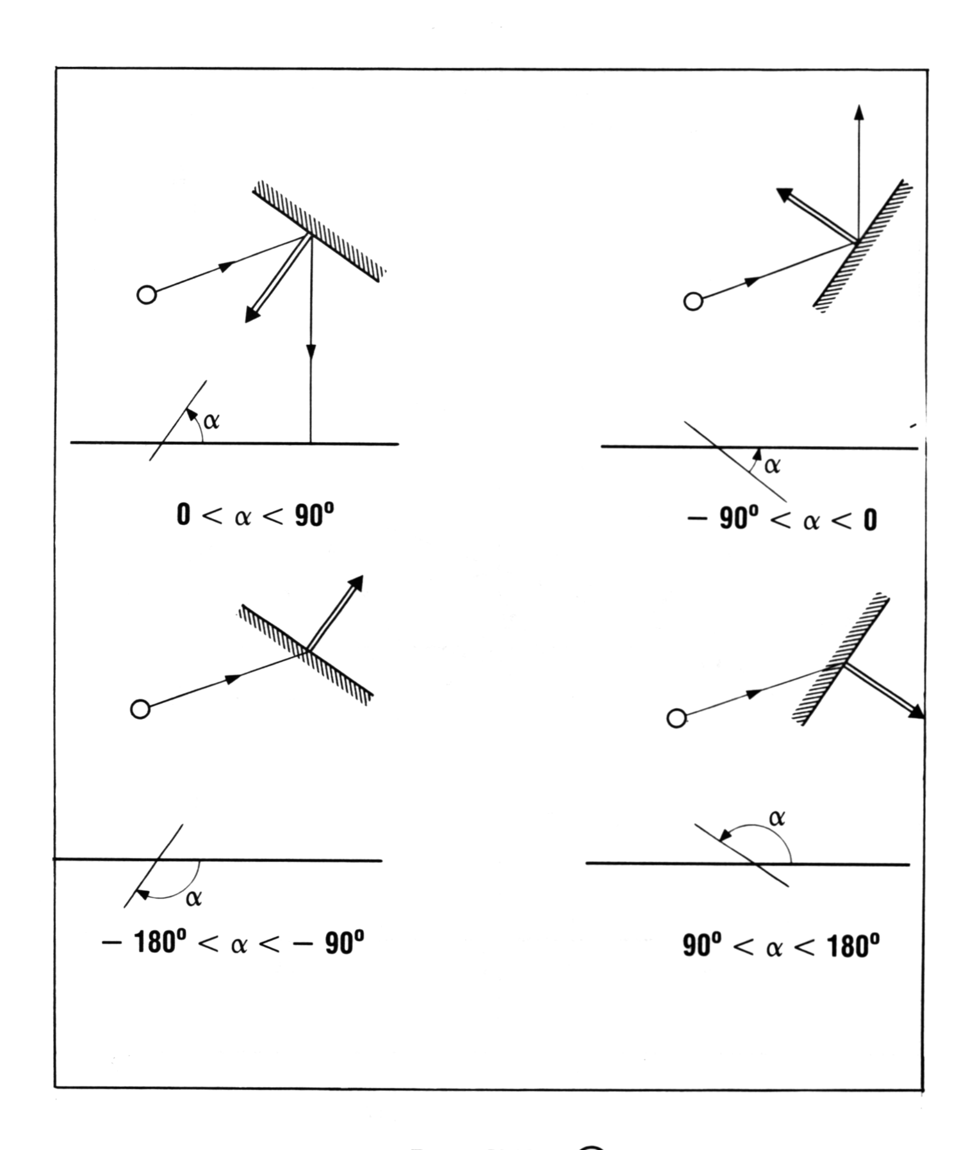


Figure VIII.A ⓒ

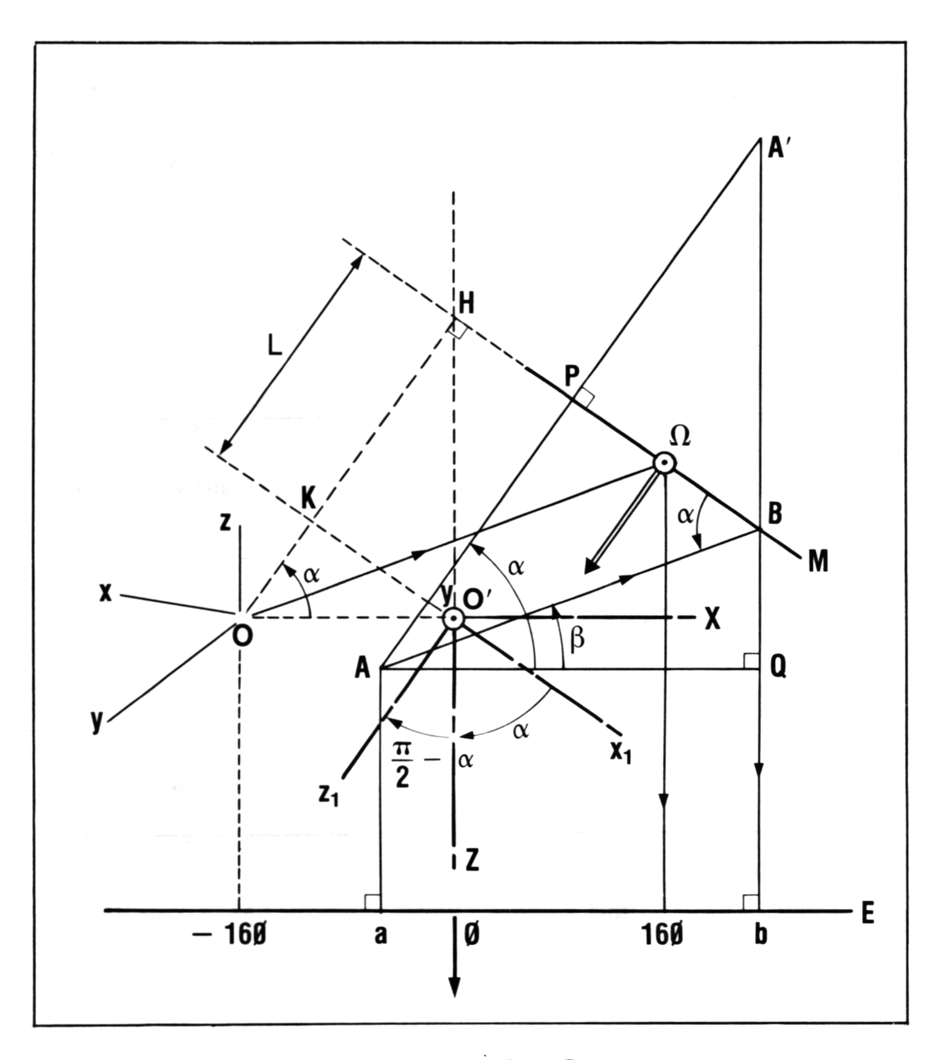


Figure VIII.A ⓓ

Le Programme VIII.1 donne un exemple de ce que peut permettre la connaissance des projections obliques. Expliquons d'abord ce qu'elles représentent à l'aide de la Figure VIII.A, schéma *a*, à gauche. Soit, dans un repère donné, une droite D qui fait, dans le plan xOz, un angle θ avec Ox. Pour tout point A de l'espace, on appelle projection de A sur Ox parallèlement à la droite D, le point *a* du plan yOx qui est tel que le

segment Aa est parallèle à D. Si D est perpendiculaire à yOx, nous retrouvons la projection orthogonale habituelle. Cela est confirmé par la matrice de cette projection, qui s'écrit :

$$\begin{bmatrix} 1 & 0 & -\operatorname{cotg}(\theta) \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Rappelons en effet que la cotangente est l'inverse de la tangente, et vaut 0 pour un angle droit. Dans ce cas, on retrouve la matrice de projection déjà citée. A quoi servent ces projections obliques ? Il y a deux utilités principales. D'une part, si le soleil se trouve dans la direction pointée par la droite D, et si on a en A une petite bille par exemple, eh bien, en a se trouve l'ombre de la bille sur le sol (plan xOy). Ainsi, les projections obliques permettent de *dessiner l'ombre d'un objet*, même lorsque le soleil n'est pas au zénith.

C'est une autre utilisation, plus particulière mais aussi plus fine, que nous allons développer à présent. Sur le schéma a, à droite, on a dessiné un objet quelconque, l'écran E vu de dessus, et un miroir M vu aussi de dessus, ce miroir étant vertical comme E, et sa normale inclinée d'un angle α avec E. Le dispositif est représenté plus clairement sur le schéma b (toujours de la Figure VIII.A). Vous constatez ainsi que sur l'écran, on peut voir non seulement la projection normale de l'objet sur la moitié gauche, mais en outre, sur la moitié droite, la projection du reflet de l'objet dans le miroir...

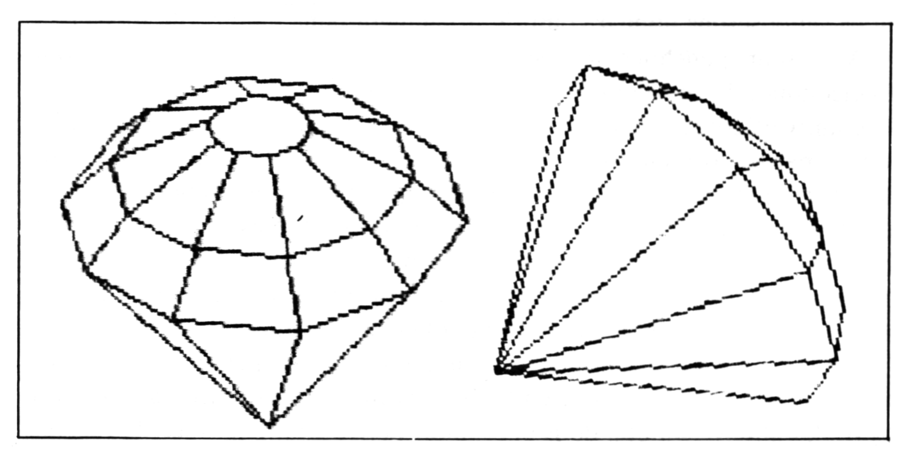


Figure VIII.B : Un polyèdre... et son reflet.

Si le miroir avait été choisi horizontal au lieu de vertical, on aurait pu avoir un effet d'objet se mirant dans une flaque d'eau. Si on noircissait tout le contenu du reflet, on verrait alors l'ombre portée par l'objet sur un mur, la lampe étant placée dans la direction de la droite D du schéma *a*, à droite. Comme ce sont des cas plus simples (qui peut le plus peut le moins), je vais vous expliquer comment obtenir effectivement sur votre écran à la fois l'image de l'objet et son reflet.

Les lois de l'optique affirment que, lorsqu'un rayon lumineux est dévié par un miroir, comme le sont ceux du schéma *a*, ils font le même angle i avant de ne se réfléchir qu'après. Il suffit de le savoir.

Notez au passage que i est égal à $90 - \alpha$ (en degrés). Il en résulte que β (voir schéma) vaut $90 - 2i$, soit $2\alpha - 90$. Toutes ces relations angulaires se déduisent de la figure. Au passage, je vous conseille donc de ne jamais omettre de faire une figure bien propre avant un raisonnement de géométrie, faute de quoi on ne s'en sort jamais.

Avant de rentrer dans le détail des calculs, je dois parler un peu de l'angle α. Un miroir, chacun le sait, a un côté réfléchissant et l'autre mat. Sur le schéma *c*, ce dernier a été grisé. Le vecteur normal au miroir est celui qui sort par la partie réfléchissante. Ce schéma *c* représente les quatre positions types du miroir selon la valeur de α par rapport aux angles 0, 90°, − 90°, 180° et − 180° (à 360° près comme toujours pour des angles). On voit que c'est seulement lorsque α est compris entre 0 et 90° qu'il y a un reflet visible sur l'écran. Autrement, suivant les cas, soit il n'y a pas de reflet du tout, le miroir présentant sa partie mate, soit le reflet se trouve du côté opposé à l'écran et est donc invisible (en haut à droite). Dans la suite, nous supposerons donc α compris strictement dans l'intervalle [0,90°].

A présent, penchons-nous sur le complexe schéma *d*. Soit A un point quelconque, de coordonnées XX, YY, ZZ dans le repère O′, X, Y, Z de l'écran. Ces coordonnées se déduisent de x, y, z, coordonnées du repère Oxyz, par les formules de rotation habituelles, auxquelles il faut toutefois ajouter l'opération XX = XX − 160 (ligne 1215 du Programme VIII.1). En effet, comme nous voulons représenter à la fois le polyèdre et son reflet, mais pas superposés, il faut centrer le polyèdre sur la partie gauche de l'écran, et centrer le miroir sur la partie droite. Il existe une infinité de rayons issus de A qui viennent frapper l'écran E. Mais, de tous ces rayons lumineux, un seul nous intéresse, celui qui revient perpendiculaire à E. Ce rayon est dévié par le miroir en B. Il faut, au total, connaître la différence entre les abscisses b et a de B et A, pour pouvoir faire le tracé de B. Pour cela, il existe plusieurs méthodes. L'une d'entre elles consiste, après s'être

mis dans le système de coordonnées adapté au miroir x_1, y_1, z_1, qui se déduit de O′, XYZ par une simple rotation d'angle $\pi/2 - \alpha$ autour de O′ Y (voir figure), à faire la projection oblique dont j'ai déjà parlé. Attention toutefois, car dans ce système de coordonnées le miroir a pour équation $z_1 = \mathrm{L}$, où L est la distance de O′ à M, ou encore la distance KH. A propos de L, notez que, comme H est à la verticale de O′ (ce n'est pas un hasard de figure, et j'expliquerai tout à l'heure pourquoi), et comme OO′ vaut 160 et qu'enfin l'angle $\widehat{\mathrm{O'OH}}$ est α, on a les relations suivantes :

$$\mathrm{OH} = \frac{160}{\cos(\alpha)} \quad \text{et} \quad \mathrm{OK} = 160\cos(\alpha)$$

d'où

$$\mathrm{L} = \mathrm{KH} = 160\left(\frac{1}{\cos(\alpha)} - \cos(\alpha)\right) = \frac{160\sin^2(\alpha)}{\cos(\alpha)}$$

L est ainsi calculé. On peut donc trouver les coordonnées de B.

Une fois n'est pas coutume, je voudrais exposer ici un calcul géométrique de la distance de *b* à *a*. Il y a là une arrière-pensée. Si, en effet, vous cherchez à refaire le même travail de miroir en perspective optique (voir Chapitre X), vous ne pourrez plus utiliser les projections obliques. Par contre, vous pourrez utiliser une loi de l'optique qui affirme que, lorsque vous regardez dans un miroir, vous semblez voir le symétrique de l'objet reflété par rapport au plan du miroir. Ainsi, placé devant l'écran, vous semblez voir en *b* le point A′, symétrique de A par rapport à M, et qu'on appelle de ce fait son *image*. Ce n'est donc pas réellement B qui doit être dessiné sur l'écran, mais A′. En fait, il se trouve qu'ici ces deux points se projettent tous les deux en *b*. Néanmoins, cette remarque permet de calculer $b - a$ sans trop de problèmes. En effet, A′ étant le symétrique de A, le point P est à la fois le milieu de A′ et A, et la projection (orthogonale cette fois) de A sur M. Dès lors A′A = 2 AP. Or $\mathrm{AP} = z_1 + \mathrm{L}$, où z_1 est la troisième coordonnée de A dans le système d'axes dont j'ai parlé tout à l'heure. Enfin, l'angle $\widehat{\mathrm{PAQ}}$ vaut α. Donc $b - a = \mathrm{AQ}$ vaut $2\,(z_1 + \mathrm{L})\cos(\alpha)$. Nous avons calculé L tout à l'heure, et z_1 vaut $\mathrm{ZZ}\sin(\alpha) - \mathrm{XX}\cos(\alpha)$. Après transformation, et compte tenu des valeurs des sinus et cosinus de l'angle double obtenu en Chapitre II, on obtient la formule de la ligne 1250 du programme.

Notez que si A était en O, il faudrait que *b* vaille 160 pour que le miroir soit bien centré. C'est dans ce but que le miroir est déplacé vers l'arrière ou vers l'avant par rapport au polyèdre, selon la valeur de l'angle α. De ce

fait, le symétrique de O par rapport à M (non marqué sur la figure) est à la verticale de Ω (homologue de B dans le cas A = O), et H est au milieu de l'écran, ce que j'avais affirmé tout à l'heure.

Donc, lorsqu'on *tourne* le miroir (cela se fait avec les touches B et N dans le programme), celui-ci doit en outre reculer ou avancer, faute de quoi le reflet sortirait de l'écran. Mais il y a pire : il doit aussi changer de largeur, sinon le reflet sortirait... du miroir, ce qui est un comble ! En fait, dans le programme, on suppose que le miroir est extrêmement grand, c'est pourquoi on n'en dessine pas les bords à l'écran. Ainsi, on peut sans souci tracer tout le reflet à l'écran. Cela, pourtant, n'est pas obligatoire. A votre avis, comment peut-on faire autrement ? *(Question VIII.a.)*

Avant d'en finir avec ces histoires de reflets, quelques mots sur le Programme VIII.1. Dans ce programme, la pression des touches Q, W, O, P fait déplacer le polyèdre comme d'habitude, mais aussi son reflet, qui est tracé juste ensuite. Par contre, l'appui des touches B ou N fait tourner le miroir sans bouger le polyèdre. C'est pourquoi seule la fenêtre 1 (ligne 550) est vidée : c'est la partie droite de l'écran, et ainsi seul le reflet est retracé.

D'autre part, le reflet, comme l'original, prend de la place. Ce dernier est plus petit qu'avec le programme des polyèdres seul. D'où une modification des lignes 530 et 1100 à 1160. Notez que deux flags sont utilisés dans ce programme : fgM indique à la machine si c'est au reflet ou à l'original d'être tracé ; fg*a* indique si l'angle est valable. Dans le cas contraire, vous verrez l'inscription *Miroir mal orienté* apparaître à la place du reflet. Appuyez alors sur B ou N, le reflet finira par se tracer. A propos de ces touches, sachez que vous pouvez également les utiliser avec SHIFT et CTRL, comme les touches Q, W, O, P. Cela permet des rotations plus ou moins rapides du miroir.

Enfin, je répète qu'il est facile de réadapter ce programme, soit pour des tracés différents, soit pour des miroirs placés autrement (horizontaux par exemple). Rien ne vous empêche donc à présent de mirer n'importe quoi... des alouettes, par exemple !

2. PRINCIPE D'UNE SECTION. APPLICATION AUX POLYEDRES

De même que Monsieur Jourdain faisait de la prose sans le savoir, votre charcutier ignore probablement qu'en vous coupant une tranche de

jambon, il vous fait une *section* de ce morceau de porc. On appelle en effet section une sorte de tranche d'un objet, ou plus exactement l'intersection d'un objet avec un plan donné.

Faire une ou plusieurs sections d'un objet un peu compliqué peut permettre d'en accroître notablement la compréhension. Cela montre en effet l'intérieur de l'objet. Ainsi le scanner est-il un appareil qui permet d'obtenir des sections des différentes parties du corps... sans avoir à le découper réellement, heureusement.

Faire une section d'un objet n'est pas une opération très difficile en général. Toutefois, cela peut être assez long. En effet, si l'objet est une surface (ou une réunion de surfaces par extension), d'équation $f(x, y, z) = 0$, et si le plan de section a pour équation $z = a \cdot x + b \cdot y + c$, par exemple, en remplaçant z par sa valeur on obtient une équation implicite qui ne pourra, en général, être résolue que point par point, ce qui risque d'être un peu long (voir l'Annexe B à ce sujet).

Toutefois, dans certains cas, cela peut être très facile. C'est le cas notamment si la surface interceptée est un morceau de plan : l'intersection est alors un segment qui peut être tracé instantanément si l'on en connaît les extrémités.

C'est la raison pour laquelle nous allons étudier la section sur l'exemple des polyèdres, dont les faces sont effectivement des morceaux de plans.

PROGRAMME VIII.2

```
2055 IF INKEY(60)<>-1 GOTO 4000
2645 PRINT:PRINT"_ S Pour obtenir une section"
2825 PRINT:PRINT"_ S Pour obtenir une section"
4000 '==================================== SECTIONS PARALLELES A L'ECRAN
4010 LOCATE 1,1:INPUT "Constante";cste
4020 GOSUB 1000
4030 FOR t=0 TO f-1
4040 s0=0
4050 FOR u=0 TO AA(t)-1
4060 s=c1*s2*a(t,u,0)+s1*s2*a(t,u,1)+c2*a(t,u,2)-cste
4070 IF s=0 THEN GOSUB 4700:GOTO 4230
4080 IF s*s0<0 GOTO 4200
4090 s0=s
4100 NEXT u
4110 NEXT t
4120 GOTO 2000
4200 '============================================= Premier point de section
4210 uu=u-1
4220 GOSUB 4500
4230 PLOT XX,YY,1
4240 s0=s
4250 FOR v=u+1 TO AA(t)-1
4260 s=c1*s2*a(t,v,0)+s1*s2*a(t,v,1)+c2*a(t,v,2)-cste
4270 IF s=0 THEN u=v:GOSUB 4700:GOTO 4420
4280 IF s*s0<0 THEN uu=v-1:u=v:GOTO 4400
4290 s0=s
4300 NEXT
```

```
4310 u=AA(t)-1:uu=0
4400 '======================================== Deuxieme Point de section
4410 GOSUB 4500
4420 DRAW XX,YY
4430 GOTO 4110
4500 '====================================== Calcul du Point de section
4510 quotient=c1*s2*(a(t,u,0)-a(t,uu,0))+s1*s2*(a(t,u,1)-a(t,uu,1))+
c2*(a(t,u,2)-a(t,uu,2))
4520 IF ABS(quotient)<0.001 THEN k=0 ELSE k=s/quotient
4530 x=a(t,u,0)-k*(a(t,u,0)-a(t,uu,0))
4540 y=a(t,u,1)-k*(a(t,u,1)-a(t,uu,1))
4550 z=a(t,u,2)-k*(a(t,u,2)-a(t,uu,2))
4560 GOSUB 1200
4570 RETURN
4700 '====================== Cas Particulier:sommet sur le Plan de section
4710 x=a(t,u,0)
4720 y=a(t,u,1)
4730 z=a(t,u,2)
4740 GOSUB 1200
4750 RETURN
```

Le Programme VIII.2 est, comme le précédent et le suivant, destiné à être rajouté au programme de création de polyèdres à répétition cylindrique. Cela fait, l'appui de la touche S permet de passer au calcul d'une section et à son tracé.

Plutôt que rentrer quatre constantes, il suffit ici d'en rentrer une seule, la section se faisant alors automatiquement parallèlement à l'écran, par le plan d'équation ZZ = cste. L'entrée de la constante permet de faire des sections successives. Pour faire des sections en des endroits variés du polyèdre, il suffit de le tourner suivant la bonne direction, par la méthode habituelle.

Comment la machine réalise-t-elle le tracé de cette section ? C'est assez simple. Tout d'abord, je rappelle que si l'on a un point de coordonnées (a, b, c) par exemple, et un plan d'équation $ux + vy + wz + k = 0$, il suffit de calculer $s = ua + vb + wc + k$ pour savoir si le point est en avant (s positif par exemple), en arrière (s négatif) ou sur le plan ($s = 0$ dans ce dernier cas).

Dès lors, prenons une face du polyèdre. Connaissant l'équation du plan de section (ici u, v, w sont les coordonnées de la normale à l'écran, et $k = -$ cste), nous pouvons calculer s pour tous les points de la face. Si, pour chaque point de la face, s est du même signe que pour le point précédent (s_0 dans le programme), ce qui se traduit par $s \cdot s_0$ positif, alors la face est tout entière d'un même côté du plan ; elle n'est pas coupée.

Si, par contre, on finit par avoir $s \cdot s_0$ négatif, c'est que s a changé de signe, donc que le dernier point de calcul, appelons-le A, est du côté

opposé au plan de section du point précédent B. Dès lors, le segment AB est coupé par le plan de section. Il faut trouver le point d'intersection. Comment faire ? C'est très simple. Ce point étant sur la droite AB, il existe nécessairement un nombre k tel que $x = a_0 - k \cdot (a_0 - b_0)$, etc., a_0, etc. désignant les coordonnées de A, b_0, etc. désignant celles de B, et x, y, z celles du point recherché. Pour trouver k, il suffit de se rappeler que le point est également sur le plan de section. Cela nous donne une équation, et l'on peut ainsi trouver k par un simple quotient, ce qui se fait aux lignes 4500 et suivantes dans le programme.

Ce premier point de section trouvé, il faut en chercher un autre car j'ai dit que la section d'une face par un plan est (en général) un segment : il en faut donc la seconde extrémité. Pour cela, il suffit de repartir du point A et de continuer notre test. Lorsqu'enfin on a trouvé un second produit $s \cdot s_0$ négatif, il suffit de répéter les mêmes opérations pour trouver le second point d'intersection, puis de tracer une ligne entre les deux avant de passer à la face suivante. Si aucun point ne donne un produit $s \cdot s_0$ négatif, arrivé au dernier, le programme suppose que le plan de section passe entre le dernier point de la face et le premier, et continue. Ce faisant, il omet un cas particulier qui est alors mal traité. Lequel et, à votre avis, comment peut-on corriger ce défaut ? *(Question VIII.b.)*

Précisons que le programme traite les cas particuliers où l'un des sommets de la face se trouve dans le plan de section, ainsi que celui ou l'intersection est l'une des arêtes de la face. Ce dernier détail n'est pas évident mais c'est la vérité, réfléchissez bien.

Il ne vous reste plus qu'à tester ce petit programme avant de passer à plus exaltant, les coupes.

3. PRINCIPE D'UNE COUPE. APPLICATION AUX POLYEDRES

Les sections sont faciles à faire, mais elles ont l'inconvénient d'un léger manque de clarté ; il est parfois difficile d'y reconnaître les formes de l'objet d'origine.

Ce n'est pas le cas des coupes, dont l'usage, dans l'ensemble des techniques, est beaucoup plus répandu. En contrepartie, elles sont un peu plus difficiles à réaliser.

Pour expliquer de quoi il s'agit, je vais me permettre de faire une nouvelle fois référence à votre charcutier (si vous en avez un). Lorsque celui-ci découpe une tranche de jambon, vous avez, avec la tranche, une section du jambon. Avec ce qu'il en reste, vous en avez une coupe. Une *coupe* est donc un morceau tout entier d'un objet que l'on suppose coupé en deux par un plan, alors que la section n'en est qu'une infime tranche.

La coupe permet non seulement de voir l'intersection de l'objet avec le plan de coupe, comme la section, mais en outre ce qui reste de l'objet, derrière le plan, comme si vous aviez effectivement coupé l'objet, d'où son nom. De ce fait, il est plus facile, lorsque la coupe a été réalisée, de se représenter l'objet original tout entier, et de mieux situer le plan de coupe par rapport à celui-ci. Les renseignements apportés sont ainsi plus accessibles.

Lorsqu'un architecte présente le plan d'une maison, il en présente en fait une coupe, le plan de coupe étant parallèle au plancher. Le plafond enlevé, on peut voir l'agencement des murs, voire des meubles, etc.

La coupe est donc l'instrument d'investigation le meilleur, et peut aussi bien être utilisée dans des usages très sérieux et professionnels que dans de simples jeux, pourvu qu'il s'y trouve un décor un peu compliqué.

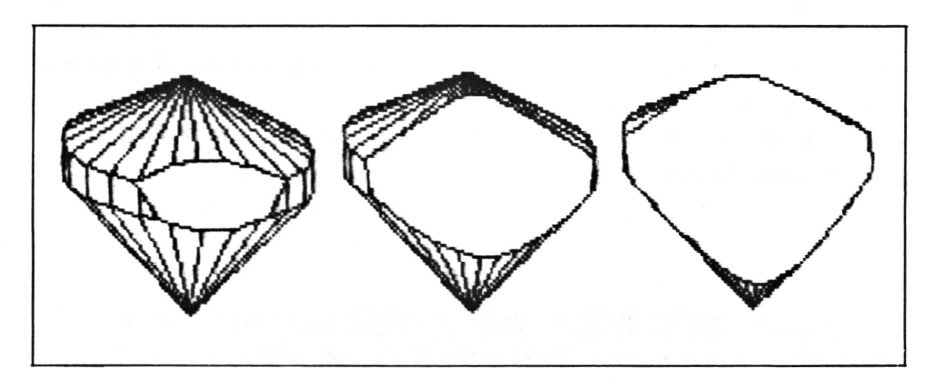


Figure VIII.C : Coupes successives d'un polyèdre. Constantes 3-1.5-0.

Les coupes ne sont pas très difficiles à réaliser sur ordinateur et je souhaite vous en convaincre sur le même exemple que pour les sections ; le Programme VIII.3 est donc également destiné à être rajouté au programme de polyèdres déjà cité.

PROGRAMME VIII.3

```
627 IF f9C<>0 GOTO 5500
730 ON f9C+1 GOTO 2000,5030
2070 IF INKEY(62)<>-1 GOTO 5000
2080 GOTO 2010
2655 PRINT:PRINT"_ C Pour obtenir une coupe."
2835 PRINT:PRINT"_ C Pour obtenir une coupe."
3315 DIM c(smax):DIM fs(f-1,2,1)
5000 '======================================== COUPES PARALLELES A L'ECRAN
5010 LOCATE 1,1:INPUT "Constante ";cste
5020 f9C=1:AAs=0:GOTO 600
5030 f9C=0
5040 FOR u=0 TO AAs-1
5050 x=fs(u,0,0):y=fs(u,1,0):z=fs(u,2,0)
5060 GOSUB 1200:PLOT XX,YY,1
5070 x=fs(u,0,1):y=fs(u,1,1):z=fs(u,2,1)
5080 GOSUB 1200:DRAW XX,YY
5090 NEXT
5100 GOTO 2000
5500 '======================================== Routine Principale de coupe
5510 umax=AA(t)-1:f9=0
5520 FOR u=0 TO umax
5530 c(u)=c1*s2*a(t,u,0)+s1*s2*a(t,u,1)+c2*a(t,u,2)-cste
5540 IF c(u)*c(0)<0 THEN f9=1
5550 IF c(u)<0 THEN u0=u:t0=t
5560 NEXT
5570 IF c(0)<=0 AND f9=0 GOTO 630
5580 IF f9=0 GOTO 720
5590 FOR u=1 TO umax
5600 IF c(u)*c(0)<0 GOTO 5620
5610 NEXT
5620 umin=u:uu=0
5630 FOR u=umin+1 TO umax
5640 IF c(u)*c(umin)<0 THEN uu=u:GOTO 5660
5650 NEXT
5660 IF uu=0 THEN uuu=umax ELSE uuu=u-1
5670 IF c(umin)<=0 THEN u1=umin:u2=uuu:u3=uu:u4=umin-1 ELSE u1=uu:u2=umin-1
:u3=umin:u4=uuu
5680 IF u2<u1 THEN v2=u2+umax+1 ELSE v2=u2
5690 IF Prod<=0 GOTO 5760
5700 FOR v=u1 TO v2
5710 IF v>umax THEN u=v-umax-1 ELSE u=v
5720 x=a(t,u,0):y=a(t,u,1):z=a(t,u,2)
5730 GOSUB 1200
5740 IF v=u1 THEN PLOT XX,YY,1:a=XX:b=YY ELSE DRAW XX,YY
5750 NEXT
5760 uv=u2:ui=u3:l=0
5770 GOSUB 6000:IF Prod>0 THEN DRAW XX,YY
5780 uv=u1:ui=u4:l=1
5790 GOSUB 6000:IF Prod>0 THEN DRAW XX,YY:DRAW a,b
5800 AAs=AAs+1
5810 GOTO 720
6000 '============================================ Calcul du Point de coupe
6010 quotient=c1*s2*(a(t,uv,0)-a(t,ui,0))+s1*s2*(a(t,uv,1)-a(t,ui,1))+
c2*(a(t,uv,2)-a(t,ui,2))
6020 IF quotient=0 THEN k=0 ELSE k=c(uv)/quotient
6030 x=a(t,uv,0)-k*(a(t,uv,0)-a(t,ui,0))
6040 y=a(t,uv,1)-k*(a(t,uv,1)-a(t,ui,1))
6050 z=a(t,uv,2)-k*(a(t,uv,2)-a(t,ui,2))
6060 fs(AAs,0,l)=x:fs(AAs,1,l)=y:fs(AAs,2,l)=z
6070 IF Prod>0 THEN GOSUB 1200
6080 RETURN
```

Suivant le même principe que pour les sections, une coupe sera obtenue en appuyant sur la touche C. Le plan de coupe est également parallèle à l'écran, et l'équation est la même : ZZ = cste, la constante étant demandée au début, après l'appui de la touche C.

Ici encore, on procède face après face. Les valeurs que nous avions appelées *s* sont conservées au cours du traitement d'une face donnée dans une petite matrice *c*. Quant aux points de la section, qui forment une sorte de face supplémentaire au polyèdre coupé, ils sont conservés dans la matrice *fs*, pour des raisons que j'expliquerai tout à l'heure. Ces deux matrices sont dimensionnées juste après *a* (ligne 3315).

Lorsqu'on pratique une coupe, plusieurs cas peuvent se présenter suivant les faces.

1. La face n'est pas coupée : tous les $c(u)$ sont du même signe que $c(0)$ (qui sert ici — arbitrairement — de référence). Dès lors, *fg* est mis à zéro. Si $c(0)$ est de surcroît négatif, la face est tout entière visible (car on trace ce qui se trouve en *arrière* du plan de coupe) ; il ne reste qu'à la tracer normalement. Sinon, la face est en avant du plan de coupe : on ne trace rien et on passe à la suivante. Tout cela se traduit par les lignes 5500 à 5580.
2. La face est coupée, le flag fg est à 1. Cela se produit lorsque l'un des $c(u)$ a un signe opposé à $c(0)$. Cette valeur de u est appelée un temps $u_{\min}$. Comme pour les sections, on cherche alors une seconde valeur de u telle que $c(u)$ soit du signe opposé à $c(u_{\min})$. Cette valeur est appelée *uu*. Si on ne la trouve pas, c'est que tous les points de $u_{\min}$ à $u_{\max}$ (dernier numéro) sont du même côté, donc que le premier à être du côté opposé est le point numéroté 0. Dans ce dernier cas, son prédécesseur est $uuu = u_{\max}$. Sinon $uuu = uu - 1$.

On a à présent les segments coupés : ce sont ceux qui joignent les points de n^{os} $u_{\min} - 1$ et $u_{\min}$, d'une part, *uuu* et *uu*, d'autre part. Reste, suivant le signe de $c(u_{\min})$, à savoir, dans chacune de ces deux paires, lequel est le visible, et lequel le non visible (qui est en avant du plan de coupe). Cela est fait en ligne 5670. Dès ce moment, u_1, u_2, u_3 et u_4 désignent respectivement les numéros d'ordre du premier point visible, du dernier, du premier point non visible et du dernier.

On est alors presque au bout. Que faut-il faire à présent ?

Tout d'abord, tracer les segments visibles de la face, à condition que *prod* soit positif (car la face peut être visible du point de vue de la coupe, mais néanmoins cachée par les autres faces). Puis chercher, sur chacune

des deux arêtes coupées (u_1, u_4) et (u_2, u_3), le point d'intersection avec le plan de coupe, ce qui est fait de la même manière que pour les sections, par la routine des lignes 6000 et suivantes. Pour cette routine, je signale que *uv* désigne le numéro du point visible du segment coupé, et *ui* celui de l'invisible. Cela a de l'importance. Si prod est positif, encore une fois, on trace les petits morceaux visibles des segments coupés. Sinon, rien n'est tracé.

C'est seulement une fois la dernière face traitée que le programme s'oriente sur la ligne 5030 (en 730). Là, on parachève le travail en traçant la section dont les points ont été conservés dans la matrice *fs*. Puis l'on revient au clavier.

L'utilité de cette conservation des points de la section sous la forme d'une matrice n'est pas évidente. En fait, elle vise de légères extensions de ce programme que vous n'aurez, je pense, guère de mal à faire vous-même. La première permettrait de retracer ensuite la section seule. La seconde permettrait de faire pivoter le polyèdre ainsi coupé, sans le reconstituer tout de suite. Il suffirait pour cela de conserver également pour chaque face les valeurs de u_1 et u_2, s'il y a lieu.

Précisons enfin, au sujet de ce petit programme très extensible, qu'en outre il peut être rajouté aussi au programme de section (VIII.2). Vous disposeriez ainsi d'un programme permettant les deux opérations à la fois (les numéros de lignes ont été spécialement prévus à cet effet). Enfin, il ne tient qu'à vous de l'adapter, comme le précédent, à d'autres objets que les polyèdres. Cela sera certainement moins difficile que la création même de ces autres objets.

REPONSES AUX QUESTIONS DE CE CHAPITRE

Question VIII.a

On peut tracer les bords du miroir (ils ont pour abscisses $160 - K \cdot \sin(\alpha)$ et $160 + K \cdot \sin(\alpha)$, où 2 K est la largeur du miroir). Dès lors, si le reflet déborde (abscisse d'un point non comprise entre ces deux bornes), il faut calculer l'intersection du segment théorique avec le bord du miroir, et ne tracer que jusqu'à ce point. L'image n'est pas alors entière.

Questions VIII.b

On omet le cas où un seul point est commun à la face et au plan de section. C'est alors obligatoirement un sommet. Pour corriger ce défaut, il suffit de calculer s au point $u = 0$ en ligne 4310. Dès lors, si $s \cdot s_0$ est négatif, faire comme en ligne 4310 et aller en 4400, sinon passer directement à la face suivante sans rien tracer de plus.

IX

LA NAPPE A CARREAUX : TRACE D'UNE SURFACE QUELCONQUE EN TROIS DIMENSIONS

Nous avons vu au Chapitre VII comment tracer un morceau de surface polynomiale. Nous allons à présent chercher à tracer une surface absolument quelconque, mais pas de la même façon. Ici, il s'agira de tracer un quadrillage porté par la surface et qui la fera ainsi apparaître, et non d'en tracer l'horizon, chose que nous savons d'ores et déjà faire en théorie, et rarement exécuter en pratique (car le calcul de dérivées partielles quelconques est extrêmement difficile, voire impossible sur ordinateur).

L'intérêt de ces surfaces quelconques n'est pas seulement "l'amour de l'art". Sous la forme $z = f(x, y)$, elles permettent de fabriquer très facilement des paysages splendides, qui peuvent fournir d'innombrables décors à des jeux extrêmement variés. C'est pourquoi l'étude de ces surfaces (à peine plus particulières que les autres en vérité), et notamment de leurs parties cachées, fera l'objet des trois derniers paragraphes de ce chapitre.

1. DEFINITION — PRINCIPES

J'ignore si vous avez chez vous une grande nappe avec des carreaux, mais dans le cas contraire rien ne vous empêche de l'imaginer.

Si vous posez une telle nappe sur une surface quelconque (votre table, par exemple), elle en épousera les formes, et les carreaux, en principe bien droits, se déformeront également. De ce fait, lorsqu'on verra ce spectacle de loin, les lignes de la nappe épousant les plis et replis de la surface permettront de se faire une idée très précise de cette surface que, par contre, on ne verrait pas très bien si elle se trouvait blanche et nue ; dans ce cas, seuls les jeux d'ombre et de lumière permettraient de voir les bosses et les creux. Or ces jeux n'existent pas sur votre ordinateur, où ils sont fort difficiles à mettre en œuvre (voir chapitre précédent).

Par contre, il n'est pas très difficile de faire tracer sur une surface des lignes semblables à celles d'une nappe à carreaux, et l'effet tridimensionnel obtenu est souvent saisissant ; les figures de ce chapitre sont là pour en témoigner.

Comment faire ? Ce n'est pas très compliqué pourvu que l'on s'entende sur les définitions. Car, en fait, qu'est-ce qu'une surface ? Nous avons jusqu'ici considéré la notion comme intuitive (elle l'est), mais nous allons néanmoins préciser un peu cette intuition.

Dorénavant, le terme de surface désignera pour nous l'ensemble des points dont les coordonnées (x, y, z) (ou les coordonnées cylindriques, ou sphériques) obéissent à une équation du type $f(x, y, z) = 0$. La fonction f peut être à peu près n'importe quoi, pour nous qui ne souhaitons pas nous encombrer de subtilités mathématiques. De toute façon, cette fonction devra s'exprimer avec les fonctions mathématiques de votre Amstrad. Par exemple, l'équation $2x^2 + y^2 - \cos(z) - 1 = 0$ représente une certaine surface, non polynomiale à cause du cosinus.

Cette nappe pourrait être représentée directement avec le système des équations implicites, mais vous y seriez encore dans un mois ! Il y a en fait plus simple, car cette surface est aussi une nappe paramétrée.

Ce nom de *nappe paramétrée,* qui n'est pas de moi, mais fait très sérieusement partie des mathématiques, désigne une surface ayant la particularité d'être paramétrable, c'est-à-dire telle qu'il existe deux paramètres u et v, et trois fonctions A, B, C de ces deux paramètres, afin

Figure IX.A

que $x = \mathrm{A}(u, v)$; $y = \mathrm{B}(u, v)$; $z = \mathrm{C}(u, v)$ soit équivalent, pour toutes valeurs de u et de v, à $f(x, y, z) = 0$.

Nous connaissons déjà cette histoire de paramètre pour les courbes du plan. Ainsi, souvenez-vous que le cercle qui a pour équation $x^2 + y^2 - 1 = 0$ peut être tracé sur votre ordinateur en écrivant $x = \cos(t)$, $y = \sin(t)$, et en faisant varier t de 0 à 360°. Dans ce cas, il n'y a qu'un seul paramètre, t, parce qu'il s'agit d'une courbe ; pour une surface, il en faut deux, et il y a trois coordonnées. Ne vous interrogez pas sur la signification de ces paramètres : ils n'en ont généralement aucune. Ce sont de simples intermédiaires de calcul, contrairement aux coordonnées qui, elles, représentent effectivement la position d'un point de la surface.

Dans l'exemple donné, on s'aperçoit que si l'on pose $x = \cos(u) \cdot \cos(v)$ et $y = \sqrt{2} \cdot \cos(u) \cdot \sin(v)$, on trouve $2x^2 + y^2 = 2\cos^2(u)$. Étant donné l'équation, on en déduit que $\cos(z) = 2\cos^2(u) - 1 = \cos(2u)$. On peut donc poser $z = 2u$. On a alors les trois fonctions A, B, C. Cette démonstration semble tirée d'un chapeau de magicien, et de fait elle l'est. En effet, le problème du passage de l'équation dite intrinsèque $f(x, y, z) = 0$ aux équations dites paramétriques $x = \mathrm{A}(u, v)$, etc. est théoriquement toujours possible, mais en pratique fréquemment infaisable, et de même du passage réciproque. L'exemple ici était spécialement conçu.

Nous supposerons à l'avenir que l'on a su trouver les équations paramétriques sur la surface, ou plutôt que l'on s'est donné la surface avec des équations paramétriques, car l'équation intrinsèque, malheureusement, n'est pas d'un bien grand usage (voir néanmoins les Annexes A et B).

Partant des équations paramétriques, par contre, tout est simple. Rappelons tout d'abord qu'on ne peut tracer la surface en soi : on n'obtiendrait guère sur l'écran qu'une grosse tache, voire un écran entièrement lumineux, ce qui ne renseigne guère.

Nous allons tracer des courbes qui sont sur la surface, comme si avec un crayon on s'amusait à dessiner dessus. Les ondulations des ces courbes mettront en valeur le relief de la surface.

Quelles courbes tracer ? Les plus simples possible. Remarquez tout d'abord que, dans l'espace, une courbe peut aussi avoir des équations paramétriques, comme dans le plan, du type $x = a(t)$; $y = b(t)$; $z = c(t)$, où a, b, c sont des fonctions du paramètre t (rappelez-vous : un seul paramètre pour une courbe, deux pour une surface). Dès lors, remarquons que si l'on impose par exemple à u d'avoir une valeur fixe u_0, on peut

appeler a, b, c les fonctions de v définies par : $a(v) = A(u_0, v)$, etc. Par exemple, si $u_0 = 0$, et avec les fonctions que nous avions précédemment :

$x = \cos(u) \cdot \cos(v)$ devient $x = \cos(0) \cdot \cos(v) = \cos(v)$

$y = \sqrt{2}\cos(u) \cdot \sin(v)$ devient $y = \sqrt{2}\sin(v)$

$z = 2u$ devient $z = 0$

x, y, z ne sont plus les fonctions que d'un seul paramètre, l'autre étant maintenu constant. Le point se trouve donc sur une certaine courbe (ici, c'est une ellipse se trouvant dans le plan $z = 0$). Pour tracer cette courbe, il suffit de faire varier v de v_{min} à v_{max} (par exemple de 0 à 360°). Pour chaque point, on calcule x, y, z, puis XX et YY (n'oublions pas la projection), et on place un petit point sur l'écran avant de passer à la valeur suivante de v. C'est à peu près aussi simple que pour une courbe plane. Lorsque la courbe est finie, c'est u_0 que l'on change, et on recommence, u_0, c'est-à-dire u, variant entre chaque courbe de u_{min} à u_{max}. Lorsque toutes ces opérations sont finies, on obtient une succession de courbes les unes à côté des autres. Pour avoir un quadrillage, il faut des courbes dans l'autre sens : il faut alors faire un tracé à v constant.

Voici le programme que cela donnerait :

```
FOR u = umin TO umax STEP (umax-umin)/n :

FOR v = vmin TO vmax STEP h :

x = cos (u) * cos (v) : y = cos (u) * sin (v) * sqr (2) : z = 2 * u :

GOSUB "calcul de xx et yy" : DRAW XX, YY : next : next
```

puis recommencer en échangeant le rôle de u et v.

Dans ce programme, comme dans tout autre similaire, h désigne la finesse du tracé (plus h est petit, mieux les courbes sont faites, mais plus c'est long), et n le nombre de courbes sur les côtés du quadrillage (nombre entier ; plus n est grand, plus le quadrillage est petit et esthétique, mais plus le tracé est long).

Je vous propose d'essayer ce petit programme, avec des fonctions variées. Le calcul de XX et YY est toujours le même. Prenez garde en

choisissant vos fonctions, ainsi que u_{min}, etc., que la surface ne déborde pas trop de l'écran : vous risqueriez de ne plus rien voir, et en plus d'avoir un message d'erreur.

2. USAGE DES COORDONNEES NON CARTESIENNES — USAGE DES FONCTIONS MATHEMATIQUES

Nous avons jusqu'à présent parlé des coordonnées cartésiennes x, y, z, mais rien n'empêche d'en utiliser d'autres. Nous connaissons ainsi les coordonnées cylindriques (r, φ, z) et sphériques (r, ψ, θ). Si l'on écrit dans ces systèmes de coordonnées des équations paramétriques $r = \mathrm{A}(u, v)$, etc., nous obtiendrons aussi une nappe paramétrée, différente bien sûr. Cela ouvre un champ nouveau de dessins de surfaces.

Le calcul des coordonnées projetées XX, YY n'est guère plus compliqué. Avec :

$$\mathrm{XX} = (s_1 \cdot x - c_1 \cdot y) \cdot e \quad \text{et} \quad \mathrm{YY} = (s_2 \cdot z - c_2 \cdot (c_1 \cdot x + s_1 \cdot y)) \cdot e,$$

on obtient les formules suivantes :

En cylindriques :

$$\mathrm{XX} \ = e \cdot r \cdot \sin(\mathrm{psi} - \varphi)$$

$$\mathrm{YY} \ = e \cdot (s_2 \cdot z - c_2 \cdot r \cdot \cos(\mathrm{psi} - \varphi))$$

En sphériques :

$$\mathrm{XX} \ = e \cdot r \cdot \sin(\mathrm{psi} - \psi) \cdot \sin(\theta)$$

$$\mathrm{YY} \ = e \cdot r \cdot (s_2 \cdot \cos(\theta) - c_2 \cdot \sin(\theta) \cdot \cos(\mathrm{psi} - \psi))$$

Attention ne pas confondre dans ce dernier cas (ψ, θ), coordonnées angulaires cylindriques du point projeté, et (psi, thêta), coordonnées angulaires de la direction d'observation.

Le programme (ou plutôt l'esquisse de programme) donné ci-dessus peut sans difficulté aucune être adapté à d'autres systèmes de coor-

données. Ces autres systèmes peuvent permettre d'amples variations, mais je n'y insisterai pas, pour des raisons que je donnerai au prochain paragraphe.

Avant de poursuivre, je dois vous mettre en garde contre les fonctions mathématiques utilisées trop vite sans précaution. Certaines fonctions méritent des gants parce qu'elles sont susceptibles de devenir infinies en certaines valeurs, ou assez grandes pour créer un *overflow*. Il s'agit notamment du logarithme (décimal ou naturel), infini en 0, de la tangente (infinie aux angles droits), plus généralement de toute division si le dénominateur peut s'annuler, mais aussi de l'exponentielle qui croît très fortement ; ainsi l'exponentielle d'un nombre supérieur à 88 créera un *overflow*.

Ne soyez donc pas surpris si un message d'erreur apparaît, si vous n'avez pas vérifié que votre fonction ne créera pas de débordement. Dans ce but, étudiez avec soin le domaine de variation des paramètres, c'est-à-dire les valeurs données à u_{min}, u_{max} et v_{min}, v_{max}. Trop petites, vous ne verrez rien d'intéressant. Trop grandes, *overflow*.

D'autre part, n'oubliez pas que les fonctions trigonométriques sont périodiques. Lorsque dans vos fonctions un paramètre n'apparaît que sous la forme de son cosinus, son sinus et sa tangente, inutile de le faire varier sur plus de 360° : vous feriez retracer plusieurs fois la même chose. C'était le cas par exemple de v, dans l'exemple donné au paragraphe précédent.

Enfin, défiez-vous de l'arc tangente (atn) et de la valeur absolue (abs) ou, pire, de la fonction signe (sgn). Ces fonctions sont susceptibles d'engendrer des variations brutales dans certains cas, qui créeront des "marches d'escalier" dans votre surface, ou même des erreurs. A utiliser avec soin, surtout la fonction signe.

Dernier mot sur le mode trigonométrique choisi : utilisez les radians. Je sais que jusqu'à présent nous avons utilisé les degrés, mais il faut cesser de le faire quand on a des mélanges de fonctions trigonométriques et d'autres fonctions. Par exemple, si vous avez $x = \cos(u)$ et $y = \exp(u)$, x est périodique en u. Si vous vous placez en degrés, et que vous voulez tracer une période entière (vous ne pouvez de toute façon tracer la courbe en entier, elle est infinie), de 0 à 360°, vous aurez un *overflow* dès que u vaudra plus de 89. Par contre, en radians, rien de grave : de 0 à 2 pi, rien ne se passe car 2 pi = 6, 28 ..., donc pas d'*overflow*, tant s'en faut. Vous voyez sur un exemple bien innocent que les degrés peuvent provoquer des ennuis. Ces recommandations valent pour les paragraphes qui suivent.

3. CAS ESSENTIEL : z = f (x, y)

Nous admettrons qu'il n'existe aucune méthode, et surtout pas de simple, pour tracer une nappe quelconque sans parties cachées sur votre ordinateur.

C'est cependant possible en théorie. Il faudrait pour cela enregistrer pour chaque point de l'écran les différentes intersections avec la nappe. Cela exigerait probablement de dimensionner une matrice de 640 sur 200, ce qui est impossible pour cause de mémoire, mais surtout le tracé serait si lent qu'il rendrait fou d'impatience même un vieux pachyderme scléro-tique !

Par contre, une certaine sorte de nappe combine deux avantages essentiels. Premièrement on peut en gérer les parties cachées sans prendre le risque de devoir se faire enfermer. Deuxièmement, c'est elle qui a le plus d'utilité pratique. Il s'agit des nappes où x et y eux-mêmes peuvent servir de paramètres, du type $x = x$; $y = y$; $z = \mathrm{C}(x, y)$, ou encore d'équation intrinsèque $\mathrm{C}(x, y) - z = 0$. C est une fonction quelconque de deux variables, que nous appellerons f dans la suite, puisque c'est la seule fonction en présence.

Pourquoi cette nappe a-t-elle une plus grande utilité pratique que les autres ? Cela dépend évidemment de ce que l'on souhaite faire de ces nappes paramétrées. En dehors de leur intérêt esthétique, elles forment un très beau paysage pour un jeu informatique (vous en avez probablement déjà vu, sinon les figures de ce chapitre le démontrent, me semble-t-il), soit telles quelles, soit avec de petites modifications. L'une de ces modifications possibles consiste, au lieu de tracer les carreaux, à ne pas tracer du tout de courbes, mais à mettre, aux points d'intersection de ces courbes *virtuelles*, de petits caractères. Ainsi, une surface ressemblant à une montagne pourra être retracée, en mettant là où les lignes se croisaient de petits caractères en forme de sapins. On aura alors l'impression de voir une montagne couverte d'une forêt de sapins. Naturellement, les variations sur ce thème sont infinies. Pour l'application de cette idée, elle est si simple, à partir du Programme IX.1, que je n'ai pas jugé utile d'en donner le détail. Je l'ai personnellement fait en modifiant moins de dix lignes dans le Programme IX.1, alors...

Tout cela cependant ne répond pas encore à la question de savoir pourquoi elles font de meilleurs paysages que d'autres, ces nappes. C'est

Figure IX.B

assez simple. L'axe Oz représente la verticale. L'équation $z=f(x,y)$ signifie que pour une valeur donnée du couple (x,y), c'est-à-dire à la verticale d'un point donné du sol (plan xOy), il n'y a qu'un seul point de la surface. Or c'est le cas en général dans la nature : le sol, sauf gros accidents de terrain, n'a pas de replis verticaux. En un point donné d'une carte, il n'y a qu'une altitude possible : cette altitude, c'est z, et elle varie selon x et y, les coordonnées de votre carte. Donc le sol même de la nature est, en gros, une surface du type $z=f(x,y)$. Évidemment la fonction est excentrique, car la nature est capricieuse. Mais on peut toujours, avec des fonctions simples, faire des paysages assez ressemblants, comme nous le verrons au Paragraphe 5.

Ce point étant éclairci, il faut expliquer le second : pourquoi les parties cachées sont-elles plus faciles à gérer sur de telles surfaces ?

4. UNE ASTUCE SPECIALE : PARTIES CACHEES SUR DE TELLES SURFACES

Pour ne pas tracer les parties cachées de ce genre de surface, nous aurons recours à une astuce assez connue, et qui repose plus sur la logique que sur les mathématiques, pour une fois.

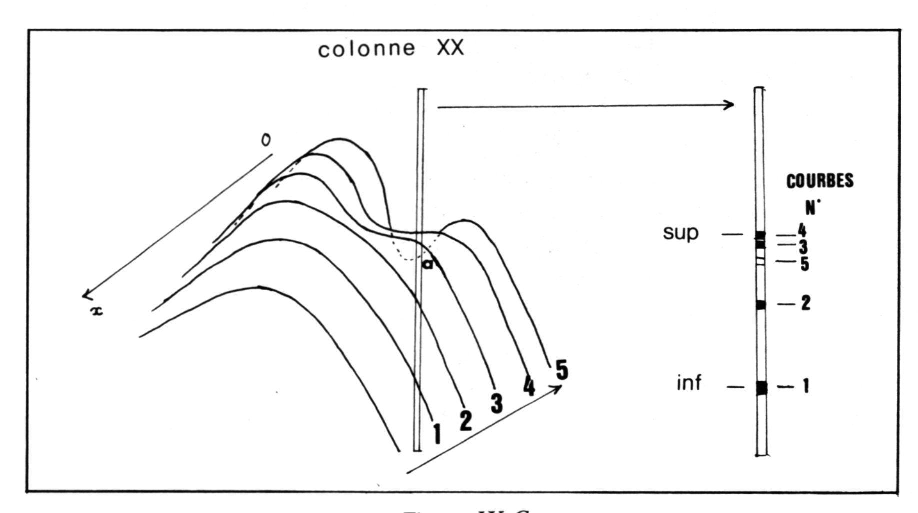


Figure IX.C

Examinons la Figure IX.C. Sur ce schéma, on a représenté, à gauche, six courbes extraites d'une surface en forme de cratère. Ces six courbes sont *parallèles*, c'est-à-dire qu'elles sont tracées à x constant. La surface est dessinée en isométrique.

Ces courbes sont tracées dans l'ordre de la flèche, c'est-à-dire dans l'ordre où x décroît. C'est essentiel. Par exemple, pour tracer la première courbe, on aura fait varier y avec $x = 2$, la deuxième avec $x = 1{,}8$, la troisième $x = 1{,}6$, etc.

Or vous remarquez que la dernière courbe passe un moment derrière les autres. Cette partie cachée est représentée en pointillé. Comment faire comprendre à l'ordinateur, lorsqu'il arrive au point a sur cette courbe, qu'il ne doit rien tracer par là ?

C'est ici que l'astuce joue. Appelons XX et YY les coordonnées (projetées) du point a. On a redessiné à droite la colonne XX. N'oubliez pas en effet que votre écran est formé de 640 colonnes, et que tout point se trouve nécessairement sur l'une d'elles.

Nous voyons que sur cette colonne il y a déjà des points des autres courbes, les cinq dernières (numérotées de 1 à 5). Parmi ces points déjà tracés sur la colonne, il en est un qui est plus haut que les autres (celui de la courbe 4), d'ordonnée sup, et un plus bas que les autres, d'ordonnée inf. Que voit-on pour YY ? YY est compris entre inf et sup. C'est pour cela que le point a est invisible. S'il était plus haut que tous les autres, ou plus bas, il serait visible.

Récapitulons ce qui s'est passé sur cette colonne depuis le début du tracé de la surface. Avant que l'on trace la courbe 1, il n'y avait jamais eu aucun point sur la colonne. Puis, la courbe 1 est tracée : appelons Y_1 l'ordonnée du point qui se trouve sur cette colonne. On pose alors sup = inf = Y_1 : l'unique point est alors le plus haut et le plus bas à la fois. Puis on trace la courbe 2. Le nouveau point est au-dessus du précédent : Y_2 est supérieur à sup. Donc le point est visible ; on pose, après l'avoir dessiné, sup = Y_2, puisque c'est lui le point le plus haut. Même chose après le tracé de la troisième courbe, puis de la quatrième. Chaque fois, le point est au-dessus des autres. A ce moment, sup = Y_4 et inf = Y_1 (n'a pas changé). Puis, sur la cinquième courbe, on arrive au point a. Et là Y_5 est compris entre inf et sup : pas de tracé, et inf et sup ne changent pas. C.Q.F.D.

Cette méthode peut paraître un peu abracadabrante, mais elle marche très bien. Pour quelle raison profonde ? En fait, tout vient de ce que j'ai déjà dit sur les courbes $z = f(x, y)$. Les colonnes de l'écran sont parallèles à

l'axe O*z*. Par conséquent, en aucun cas une des courbes tracées sur la surface ne peut couper deux fois une même colonne : dans chaque colonne, il y a au plus un point de chaque courbe, jamais deux. Et c'est cela qui fait que le "truc marche". C'est aussi la raison pour laquelle il ne saurait marcher pour une surface quelconque, ni pour la même surface si l'axe O*z* n'était pas vertical. A présent vous savez, en plus, pourquoi mes axes O*z* sont toujours verticaux. Cela donne parfois des facilités irremplaçables... outre que c'est conforme à la logique !

Notez qu'une autre chose est absolument nécessaire pour le bon fonctionnement de cette astuce, c'est le tracé avec x décroissant, c'est-à-dire avec des courbes qui semblent de plus en plus s'éloigner. Si vous refaites le même raisonnement dans le sens inverse, vous verrez qu'alors c'eût été la courbe 3 qui eût été amputée, et la courbe 5 serait complète.

Naturellement, cette astuce, il faut la répéter pour toutes les colonnes sans aucune exception. C'est pourquoi les valeurs inf et sup sont conservées dans deux matrices de 640 valeurs.

D'autre part (il faut une contrepartie à tout), l'astuce a un côté très ennuyeux. Prenons un exemple. Le programme vient de placer un point en XX0 et YY0, avec par exemple XX0 = 250. Après calcul, le point suivant sur la courbe est en XX, YY, avec par exemple XX = 245. En temps normal, il suffit de faire DRAW XX, YY, et tout est dit : un petit segment est tracé entre les deux points. Mais ici, c'est plus compliqué. Car non seulement il faut vérifier que les deux points sont visibles, en comparant d'une part YY0 avec inf (250) et sup (250), et d'autre part YY avec inf (245) et sup (245), mais il faut faire plus encore, car il y a cinq colonnes entre XX et XX0 : cinq colonnes que le segment traverse, et pour lesquelles il faut impérativement comparer toutes les valeurs intermédiaires XX*i* et YY*i* situées sur le segment (avec XX*i* entier, variant de XX0 et XX), ce pour bien marquer l'intersection de la courbe et de ces colonnes. Sinon, supposez qu'au moment du tracé de la courbe suivante, on obtienne XX = 248 un moment, avec YY supérieur à tous les autres points déjà tracés sur la colonne 248, *mais* inférieur à l'ordonnée du segment en question situé sur la colonne 248. Si l'ordonnée de ce point, qui vaut d'ailleurs YY0 + (YY − YY0) · (XX*i* − XX0)/(XX − XX0), n'a pas été enregistrée sur cette colonne, le point de la courbe suivante situé sur cette colonne sera tracé, même s'il n'aurait pas dû.

PROGRAMME IX.1

```
10 '======================================================== SURFACES EN 3D
20 '=================================================== Entree des donnees
30 MODE 1:BORDER 1:INK 0,1:INK 1,24:PEN 1
40 INPUT "Valeur minimale de x ";xmin:PRINT
50 INPUT "Valeur maximale de x ";xmax:PRINT
60 INPUT "Valeur minimale de y ";ymin:PRINT
70 INPUT "Valeur maximale de y ";ymax:PRINT
80 INPUT "Pas d'incrementation (finesse) ";h:PRINT
90 INPUT "Nombre de courbes par cote ";n:n=n-1:PRINT
100 INPUT "Facteur d'echelle";e:PRINT
110 PRINT"Voulez-vous voir tracer les axes (O/N)?"
120 WHILE INKEY(34)=-1 AND INKEY(46)=-1:WEND
130 MODE 2:INK 0,0:BORDER 0:INK 1,24:PEN 1
140 IF INKEY(46)<>-1 GOTO 200
150 '======================================================== Trace des axes
160 PLOT 320,200:DRAW 320,399:DRAWR -5,-10:PLOT 320,399:DRAWR 5,-10:
LOCATE 42,1:PRINT"z"
170 PLOT 320,200:DRAW 20,35:DRAWR 12,2:PLOT 20,35:DRAWR 8,7:LOCATE 1,23:
PRINT"x"
180 PLOT 320,200:DRAW 619,35:DRAWR -12,2:PLOT 619,35:DRAWR -8,7:
LOCATE 79,23:PRINT"y"
200 '=========================================================== Preparation
210 RAD
220 DIM inf(639):DIM sup(639)
230 FOR t=0 TO 639:inf(t)=400:NEXT
240 h=-h
300 '==================================================== Trace a x constant
310 FOR x=xmax TO xmin STEP (xmin-xmax)/n
320 IF xmax=0 THEN x1=x ELSE x1=e*x/ABS(xmax)
330 f9=0
340 FOR y=ymax TO ymin STEP h
350 IF ymax=0 THEN y1=y ELSE y1=e*y/ABS(ymax)
360 GOSUB 700:GOSUB 950
370 IF y<>ymax THEN GOSUB 800
380 XX0=XX:YY0=YY
390 NEXT:NEXT
400 '===================================================== Preparation (bis)
410 ERASE sup:DIM sup(639)
420 FOR t=0 TO 639:inf(t)=400:NEXT
500 '==================================================== Trace a y constant
510 FOR y=ymax TO ymin STEP (ymin-ymax)/n
520 IF ymax=0 THEN y1=y ELSE y1=e*y/ABS(ymax)
530 f9=0
540 FOR x=xmax TO xmin STEP h
550 IF xmax=0 THEN x1=x ELSE x1=e*x/ABS(xmax)
560 GOSUB 700:GOSUB 950
570 IF x<>xmax THEN GOSUB 800
580 XX0=XX:YY0=YY
590 NEXT:NEXT
600 GOTO 1000
700 '=========================================== Calcul de z:Points caches
710 z='fonction de x et de y'
720 XX=INT(320+90*SQR(3)*(y1-x1))
730 YY=200+90*(2*z-y1-x1)
740 IF XX>639 OR XX<0 THEN RETURN
750 I=inf(XX):S=sup(XX):IF yy>=I AND YY<=S THEN f9=0:RETURN
760 IF YY>S THEN sup(XX)=YY
770 IF YY<I THEN inf(XX)=YY
780 IF f9=0 THEN PLOT XX,YY:f9=1
790 RETURN
800 '==================================== Routine de tracage d'un segment
810 IF XX=XX0 THEN RETURN
820 IF XX>XX0 THEN pas=1 ELSE pas=-1
830 FOR XXi=XX0 TO XX STEP pas
840 YYi=YY0+(YY-YY0)*(XXi-XX0)/(XX-XX0)
850 I=inf(XXi):S=sup(XXi)
860 IF YYi>I AND YYi<S THEN f91=0:MOVE XXi,YYi:GOTO 910
865 IF YYi<=I THEN inf(XXi)=YYi
870 IF YYi>=S THEN sup(XXi)=YYi
```

```
880 IF YYi<=I THEN inf(XXi)=YYi
890 IF f91=1 THEN DRAW XXi,YYi
900 f91=1
910 NEXT
920 RETURN
950 '============================================== Validite de l'abscisse
960 IF XX>639 THEN XX=639
970 IF XX<0 THEN XX=0
990 RETURN
1000 '======================================== Sauver l'ecran sur cassette
1010 ERASE inf:ERASE sup
1020 LOCATE 1,1
1030 PRINT CHR$(7):WHILE INKEY(47)=-1 AND INKEY(53)=-1:WEND
1040 IF INKEY(53)<>-1 GOTO 1080
1050 SAVE "!Surface",B,49152,16384
1060 PRINT CHR$(7)
1070 WHILE INKEY$="":WEND
1080 MODE 1:INK 0,1:BORDER 1:PEN 1
1090 END
```

Les grands dessins valant mieux que les longs discours, je vous propose de supprimer les lignes correspondantes dans le Programme IX.1 (lignes 800 et suivantes) en les remplaçant par un simple DRAW XX, YY pour voir le désastre, si d'aventure vous n'étiez pas convaincu.

A propos de ce programme, il me reste encore quelques petites explications à donner, en vrac.

La finesse *h* désigne la distance entre deux points de calcul sur une même courbe. A choisir inférieur à 0,1, de préférence. *n* est le nombre de courbes tracées sur chaque côté du rectangle de base. Ce rectangle est celui limité par les droites $x = x_{\min}$, $x = x_{\max}$, $y = y_{\min}$, $y = y_{\max}$ dans le plan xOy. On le distingue en général très bien après le tracé. Ce rectangle est nécessaire : la plupart de ces surfaces sont infinies (sauf fonctions farfelues), on ne saurait donc les tracer tout entières.

Le programme fait en sorte que le rectangle tienne entièrement sur l'écran. Si vous voulez l'agrandir ou le diminuer, le facteur d'échelle *e* est là pour cela. Supérieur à 1, il agrandit, inférieur à 1, il diminue. Au début, rentrez $e = 1$.

Vous avez le choix entre la surface avec les trois axes du repère, ou sans. Si vous choisissez sans, appuyez fortement sur la touche *n*.

Les calculs sont faits en radians. Rien ne vous empêche d'en décider autrement, en changeant la ligne 210.

Pour chaque colonne, inf et sup sont initialisés à 400 et 0, respectivement. Ainsi le premier point d'une colonne sera toujours tracé, sans nécessiter un traitement spécial.

La finesse est donnée positive, mais comme on trace avec x et y décroissants, il faut changer le signe de h (ligne 240).

On exécute d'abord le tracé à x constant, puis à y constant. Précisons que les lignes 320 et 520 servent à faire tenir le rectangle sur l'écran. Les coordonnées projetées sont calculées en isométrique, et multipliées par 180. L'origine n'est pas placée en 320, 200, car il faudrait alors initialiser sup à -200, ce qui est assez long. On ne considère que la partie entière de XX.

La fonction $f(x, y)$ doit être placée en 710. Des exemples seront donnés au paragraphe suivant.

Deux flags sont utilisés. Ils sont destinés à savoir si l'on est au début d'une courbe (fg), ou au début d'une partie visible, après une partie cachée (fg_1). Dans ce cas, il ne faut pas faire de DRAW.

Enfin, lorsque la surface est terminée, le programme fait un bip sonore. Vous avez alors le choix entre vous arrêter (touche *f* comme fin), ou faire enregistrer la courbe sur cassette ou disquette, sous forme d'enregistrement d'écran : appuyez alors sur la barre, non sans avoir vérifié que les touches REC et PLAY sont enfoncées, car aucun message ne sera affiché (il abîmerait la surface). Lorsque l'enregistrement est fini, nouveau bip. Appuyez sur n'importe quelle touche lorsque vous serez lassé de l'admirer. Pour récupérer votre surface, ensuite, tapez LOAD "Surface", B, 49152.

Pour que vous puissiez tester ce programme sans vous casser la tête, nous allons à présent voir quelques exemples très utiles.

5. EXEMPLES ET UTILISATION

Il est rarement aisé de trouver la fonction mathématique adaptée exactement au paysage que l'on veut faire. En fait, pour peu que ce paysage soit un peu compliqué, c'est impossible sans un minimum de méthode.

On peut pourtant aboutir assez rapidement au résultat cherché. Il suffit pour cela d'utiliser la méthode des superpositions. Si par exemple vous souhaitez faire deux montagnes côte à côte, ne cherchez pas une fonction mathématique géniale qui vous donnera la bonne surface. Allez plutôt prendre dans les exemples suivants ceux qui représentent des montagnes et *additionnez*-en deux. En effet, en additionnant deux fonctions qui

représentent chacune une montagne, ou n'importe quoi, vous avez toutes les chances de voir apparaître deux montagnes sur la surface.

Quelques précautions sont toutefois nécessaires. La première est simple : les fonctions utilisées doivent être *nulles à l'infini*, c'est-à-dire que z doit devenir très faible quand on s'éloigne du centre de la fonction, centre qui sera donné dans les exemples. C'est une condition importante : si elle n'est pas réalisée, z risque de devenir trop grand à l'infini, d'où de graves problèmes. Cette condition est automatiquement réalisée pour les fonctions qui suivent, sauf pour *Déclivité*, où elle est impossible, l'un des niveaux restant forcément plus haut que l'autre.

Seconde précaution importante : s'assurer que la superposition est correcte. En effet, si l'on additionne deux fonctions dont les *centres* sont très proches, on risque d'avoir un "méli-mélo" (par exemple une grosse montagne au lieu de deux) ; il faut pour cela vérifier jusqu'où s'étend le domaine de la fonction (valeurs de x et y pour lesquelles z n'est pas faible). Si les deux domaines se superposent trop, on obtient autre chose. Cela peut être intentionnel, d'ailleurs, nous le verrons.

Voici donc les six fonctions types qui permettent de faire la majorité des paysages intéressants. Dans la suite r désigne le r des cylindriques, soit $\mathrm{sqr}\,((x-a)^2+(y-b)^2)$, où (a, b) est le centre de la fonction. u désigne une coordonnée linéaire du type : $u = \cos(\alpha) \cdot x + \sin(\alpha) \cdot y + k$. Ces six fonctions sont illustrées sur la Figure IX.D, où elles sont repérées par leur numéro d'ordre.

1. Colline - Creux

Fonction : $z = \mathrm{A} \cdot \mathrm{EXP}\,(-r^2/k^2)$

A S'il est positif, c'est la hauteur de la colline, sinon la profondeur du creux.

k Paramètre de domaine. z devient petit (inférieur à A/10) lorsque r (distance à l'axe du sommet) est supérieur à $1{,}52 \cdot k$.

2. Pic - Trou

Fonction : $z = \mathrm{A} \cdot \exp\left(\dfrac{(m-r)}{k}\right).$

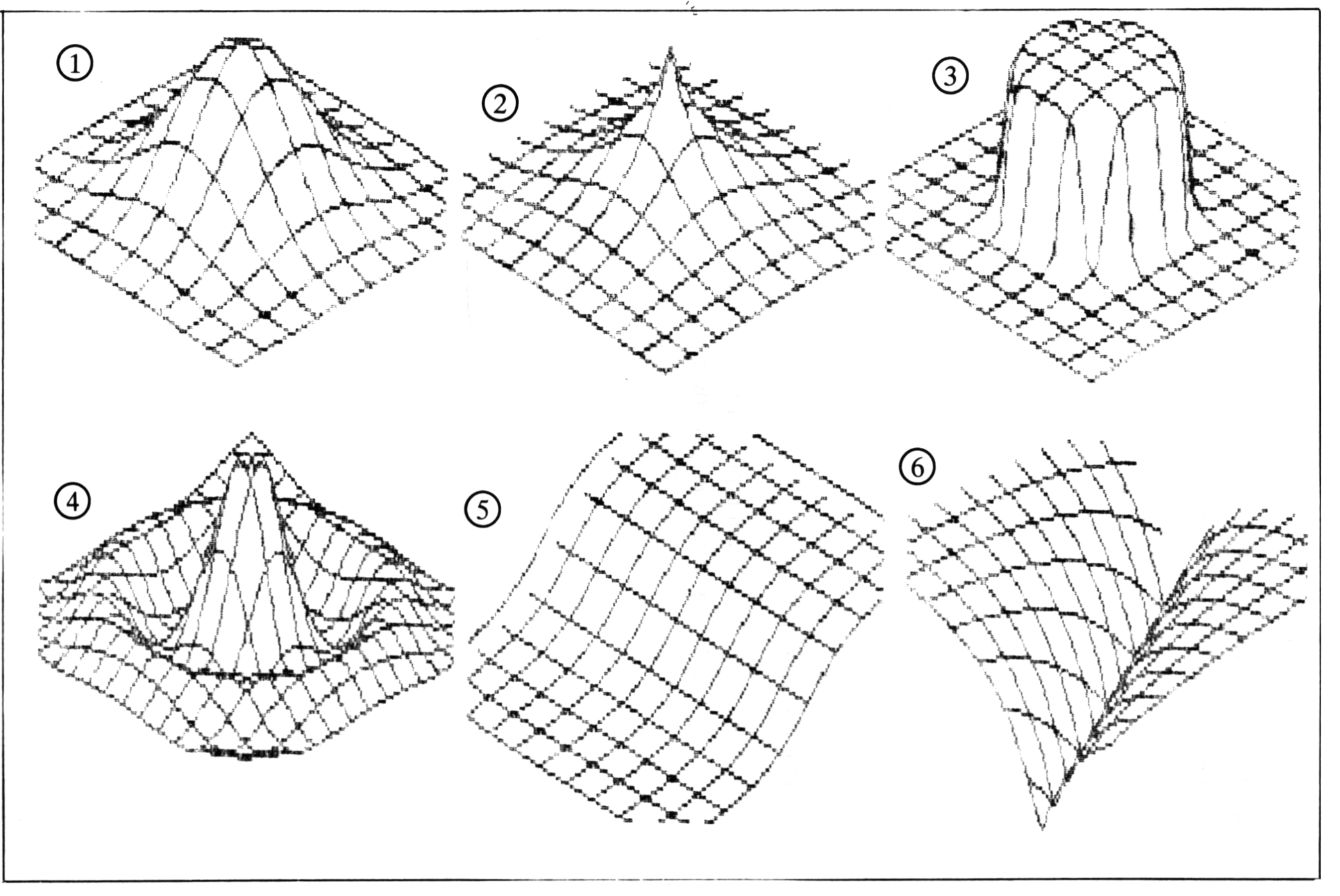


Figure IX.D

A Comme ci-dessus.

k *Positif* obligatoirement. Règle la pente du pic : plus k est important, plus le pic est évasé (*mou*, et non pointu).

m Paramètre de domaine. z devient faible (idem), lorsque r est supérieur à $m + 2{,}3 \cdot k$ (m peut être négatif).

3. Bloc rond

Fonction : $z = A \cdot (0{,}5 + ATN\ (k \cdot (m - r))/PI)$.

A Est pratiquement la hauteur du bloc.

k Plus k est important, plus les côtés du bloc sont inclinés.

m Est le rayon de la partie haute du bloc (exactement à mi-hauteur).

Domaine : Si k est beaucoup plus grand que m (au moins dix fois), le domaine s'arrête à quelques fractions de m du bloc. Ainsi, si $k = 32 \cdot m$, à une distance de $2m$ du centre du bloc, z ne vaut plus que A/100.

4. Plouf

Ainsi nommée car elle évoque l'eau quand on y jette une pierre. Mais scientifiquement, il s'agit d'un sinus cardinal.

Fonction : $Z = A \cdot \sin(rk)/rk$ (en radians).

A Hauteur du pic central.

k Règle l'élongation des ondulations (qui sont en nombre infini, mais qui sont invisibles assez vite) : plus k est petit, plus les rides sont serrées. Cette surface rappelle les ondulations du sinus qui l'engendre.

Précaution : S'arranger (avec les bornes) pour que r ne soit jamais nul, sinon *division by zero*.

Domaine : Invisible quand r dépasse quelques $10/k$.

5. Falaise - Déclivité

Fonction : $z = A \cdot ATN(u/k)/PI$ (en radians).

A Est la différence de hauteur entre les deux niveaux.

u La droite $u = 0$ est l'endroit où se trouve la déclivité.

k Paramètre de domaine : plus k est important, plus la pente est faible. Ainsi la moitié de la hauteur du niveau supérieur est atteinte quand on s'éloigne d'une distance k de la droite $u = 0$.

Conseil : Si vous souhaitez interchanger le haut et le bas, changez le signe de u, ou de k.

6. Faille

Fonction : $z = - A \cdot EXP\left(\frac{(m - abs(u))}{k}\right)$.

A, m, k Même sens que pour le pic.

u Même sens que ci-dessus.

Les figures de ce chapitre, et celle de l'introduction, ont été faites en additionnant de telles fonctions.

Ainsi pour la Figure IX.A, si l'on a, en distinguant les centres, en O : $r_0 = sqr(x^2 + y^2)$, $ra = sqr(x - 10)^2 + (y - 3)^2)$ en A (10,3), et ainsi de suite, rb, rc ... en B (8,5), en C (− 1,7), D (− 13,5), E (7, − 9), F (12, − 1), avec la fonction suivante :

$$z = \left\{ 9 \cdot EXP\left(\frac{-r_0^2}{100}\right) + 12 \cdot EXP\left(\frac{-r_0^2}{3}\right) - 8 \cdot EXP(-r_0)\right.$$

$$+ 20 \cdot EXP\left(\frac{-(1 + ra)}{2,5}\right) +$$

$$+ 15 \cdot EXP\left(\frac{-rb}{2}\right) + 25 \cdot EXP\left(\frac{-(5 + rd)}{6}\right)$$

$$+ 10 \cdot EXP\left(\frac{(-rc^2)}{4}\right) +$$

$$+ 15 \cdot \mathrm{EXP}\left(\frac{-re^2}{8}\right) - 7 \cdot \mathrm{EXP}\left(\frac{-re^2}{2}\right) + 15 \cdot \mathrm{EXP}\left(\frac{-rf}{2}\right) - 2 \Big\} / 15$$

Cette fonction est compliquée, mais elle se comprend très bien si l'on regarde seulement autour de chacun des points cités ci-dessus. Par exemple, en C, tous les termes de la fonction sont faibles, sauf $10 \cdot \exp\left(\frac{-rc^2}{4}\right)$, que l'on reconnaît comme une colline assez haute ($A = 10$) et peu évasée ($k = 2$, ce qui est faible) ; on la reconnaît sur la figure au centre légèrement à droite.

En F, seul le terme $15 \cdot \mathrm{EXP}\left(\frac{-rf}{2}\right)$ est important. On reconnaît un pic, assez haut ($A = 15$), peu évasé ($k = \mathrm{sqr}(2)$), et surtout pointu, car de domaine peu important : $m = 0$, d'où $z = \frac{A}{10}$ quand :

$r = 2{,}3 \cdot \mathrm{sqr}(2) = 3{,}3$.

On reconnaît ce pic à l'extrême gauche de la figure.

On peut ainsi identifier tous les pics de la figure.

C'est plus simple sur la Figure IX.B, de fonction :

$$z = 1{,}4 \cdot \left\{ \frac{\sin(r \cdot \mathrm{PI}/2)}{r} - \frac{3}{2} \cdot \mathrm{EXP}\left(-\frac{3r}{2}\right) \right\} \quad (r \text{ est le } r_0 \text{ ci-dessus}).$$

C'est simplement la différence entre deux fonctions : un *plouf* que l'on reconnaît bien, et un *trou* qui se voit tout en haut et au centre de ce plouf.

Le principe est plus général ; pour faire un cratère en haut d'une montagne (volcan), il faut additionner un creux ou un trou à cette montagne, avec le *même r*. C'est ce qui a été fait sur la Figure IX.A aussi, aux points O et F.

Ainsi, tous ces effets de compensation ou d'additionnement peuvent engendrer des paysages très variés sans trop de mal. Précisons que les Figures IX.A et IX.B ont été tracées en environ deux heures avec respectivement x et y variant entre -15 et 15, entre -9 et 9 ; des pas de 0,25 et 0,15 ; des échelles de 1,5 et 1 ; enfin 61 et 41 courbes par côté.

Il ne vous reste plus qu'à essayer et... à vous montrer patient.

X
LES PERSPECTIVES OPTIQUES

Ce chapitre est là pour clore cet ouvrage sur un champ nouveau d'investigations : sur la manière dont un objet est vu, et non sur la nature de l'objet. Il peut, de ce fait, s'appliquer à tout ce qui a été vu dans les chapitres précédents. Un réalisme plus frappant est toutefois obtenu au prix d'une durée plus importante de calcul (comme toujours, et c'est pourquoi l'application de ce système à l'appartement du Chapitre VII, qui constituait déjà l'un des sommets de ce livre, pour si modeste qu'il fût, ne sera qu'esquissée).

1. POURQUOI LA PERSPECTIVE OPTIQUE ?

Si vous avez eu la curiosité de faire un cube sur votre Amstrad, à l'aide du programme *Polyèdres de révolution*, vous avez peut-être été légèrement choqué par sa forme, trop parfaite, en quelque sorte : les côtés sont strictement parallèles, le fond a la même taille que la face de devant, etc.

Or, chacun sait que les objets ne nous apparaissent pas ainsi. Les côtés d'un cube ne sont pas parallèles, mais fuient vers l'horizon. Qui n'a jamais regardé, de la rue, le haut d'un immeuble : il semble moins large que le bas. Et une route qui s'enfuit vers l'horizon semble devenir de plus en plus étroite.

Les peintres de la Renaissance se sont mis en réaction par rapport à ceux du Moyen Âge, car ils voulaient en effet rendre cette réalité, que nous appellerons perspective optique, ou perspective tout court, alors que les peintres du Moyen Âge ne le faisaient pratiquement pas.

Pour cela, ces artistes durent établir les règles complexes de la perspective en peinture, basées sur une observation rigoureuse des choses vues. Ils arrivèrent ainsi à de remarquables résultats.

Je vous propose de faire la même chose. Nous allons réagir par rapport à tout ce qui précède, qui n'était pas assez bien représenté. Cela n'est pas absolument nécessaire, bien sûr, et les projections classiques que nous avons faites jusqu'à présent donnaient, dans l'ensemble, des résultats très proches de la réalité.

Cependant, dans certains cas, la perspective optique sera nécessaire, et c'est à ces cas que je pense.

2. OUI, MAIS COMMENT ? MANIERE APPROCHEE

Prenons un exemple. Vous faites un jeu où un avion se rapproche de sa cible, vous en l'occurrence, je veux dire le joueur. On doit donc voir l'avion s'approcher. Seulement, voilà. Si vous avez eu l'idée (bonne, malgré tout) d'utiliser le programme *Tous polyèdres* pour faire votre avion, un léger problème va se poser. Comment le faire s'approcher, tout d'abord ? Cela, c'est facile à résoudre, comme nous le verrons. Mais de

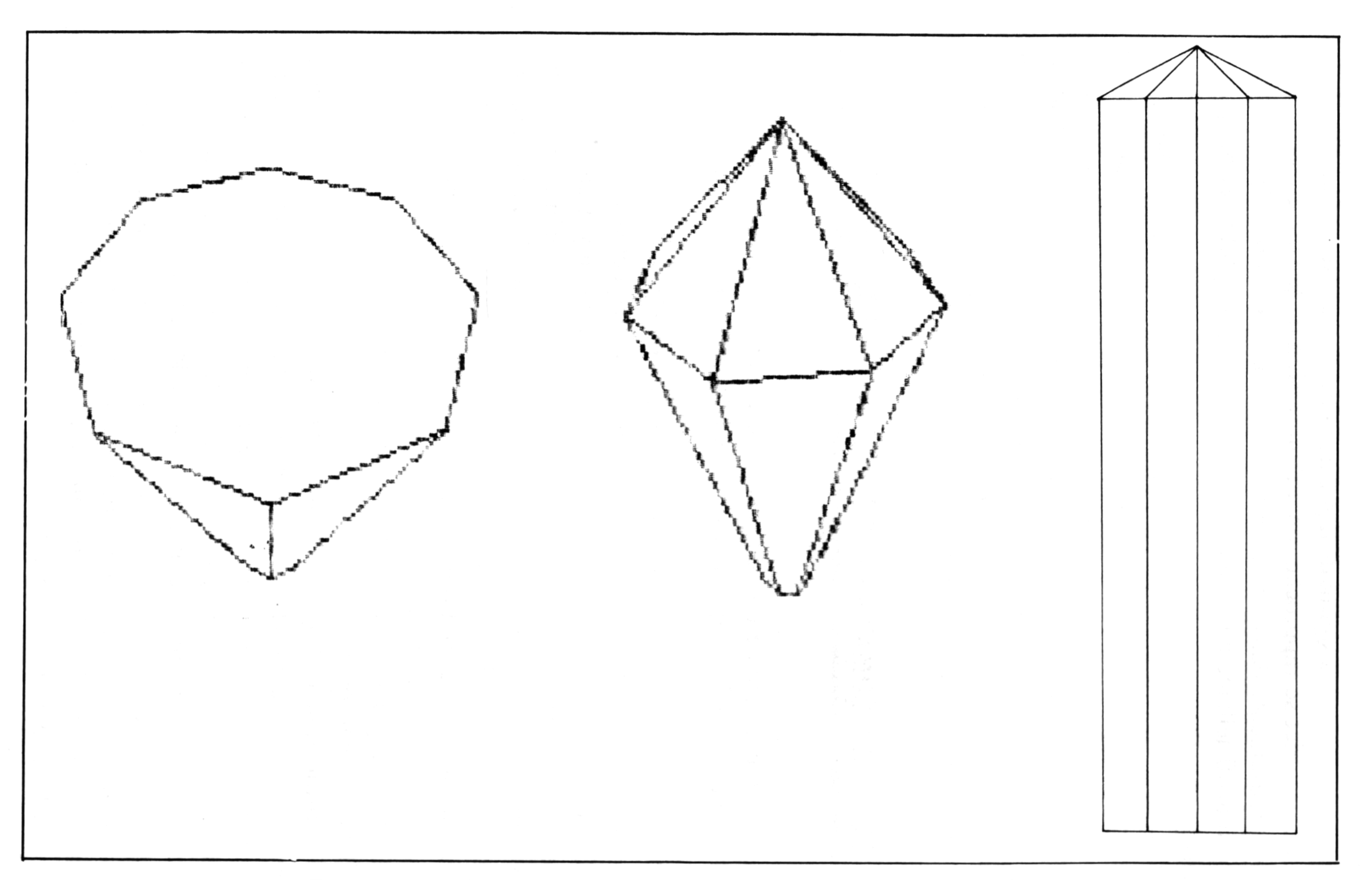


Figure X.A : Objet en perspective optique (grand angle) — Plan de l'objet.

toute façon l'avion resterait de la même taille, ce qui est très ennuyeux, car lorsqu'un objet se rapproche, on a l'impression de le voir grossir.

Heureusement, dans un tel cas, la solution est simple. Il suffit de le grossir artificiellement (et au hasard, de surcroît). L'effet n'est pas parfait, mais peut suffire.

On peut aussi procéder par approximation pour d'autres choses. Par exemple pour une route s'éloignant vers l'infini. Il faut alors espérer que le terrain est bien plat et la route bien droite, sinon tout se complique...

Par contre, pour les pièces, cela devient dommage. Combien de ces programmes de jeux vendus dans le commerce, excellents, mais péchant par une forme trop stéréotypée des pièces où le joueur se déplace... Et pourtant, ce n'est pas si difficile...

3. OUI, MAIS COMMENT ? (SUITE) MANIERE SCIENTIFIQUE

On se flatte comme on peut mais il est vrai, après tout, que c'est une démarche scientifique que nous allons adopter pour la résolution de ce problème.

Nous voyons comme les appareils photo, ou plutôt c'est le contraire. Les photos, elles, établissent automatiquement les perspectives.

Or, comment est fait un appareil photo ? En gros, il y a un objectif, et une plaque photographique derrière. Pour l'œil c'est la même chose. Dans les deux cas, l'image obtenue est inversée, comme on sait. Pour l'appareil, des lentilles auxiliaires, et pour l'œil, le cerveau, rétablissent l'image dans son sens normal. L'effet obtenu est le même que si la pellicule se trouvait devant l'objectif, comme nous le verrons.

L'effet de perspective est dû à une cause profonde qui n'est pas très difficile à comprendre.

La projection orthogonale, que nous avons utilisée jusqu'à présent, envoyait vers l'écran des rayons parallèles. La conséquence était simple : avec un tel système, une sphère de rayon dix centimètres aurait été représentée sur l'écran par un cercle de rayon dix centimètres, la projection conservant les dimensions parallèles à l'écran.

A moins d'avoir une très grande plaque photographique, je vois mal comment, avec le même système, vous pourriez photographier une telle sphère, pourtant bien plus petite qu'un simple ballon de football. Quant au mont Blanc, n'en parlons pas !...

Conclusion : votre appareil photo ne *voit* pas par une projection orthogonale. Cela, on le savait déjà. Comment fait-il alors ? C'est assez simple, en fait. Il existe dans l'appareil un point que l'on appelle focal, et que nous appellerons point O dans la suite. Au lieu que ce soit les rayons lumineux perpendiculaires à l'écran qui impressionnent la pellicule, ce sont ceux qui passent par O (n'oubliez pas qu'un objet envoie des rayons lumineux dans toutes les directions, sinon on ne pourrait pas le voir de partout).

Chaque rayon lumineux issu de l'objet et passant par O impressionne la pellicule derrière. En fait, pour nous, il s'agit essentiellement de trouver l'intersection du "rayon lumineux", c'est-à-dire en fait de la droite joignant O à l'objet vu, avec l'écran. Ce dernier, pour éviter une inversion de l'image, se trouve devant le point O, et non derrière. Voyez ainsi la Figure X.B. On y a imaginé que l'observateur (dont l'œil se trouve en O) se déplaçait dans un champ de météorites. Trois d'entre elles ont été représentées : M_1, M_2, M_3. Seules les deux dernières ont une image sur l'écran, car la droite $O - M_1$ ne touche pas l'écran. C'est là en effet une réalité : l'écran est borné, comme une pellicule photographique ou votre rétine. Vous ne pouvez voir à la fois tout ce qui vous entoure, et l'appareil photo ne peut prendre tout le paysage d'un coup.

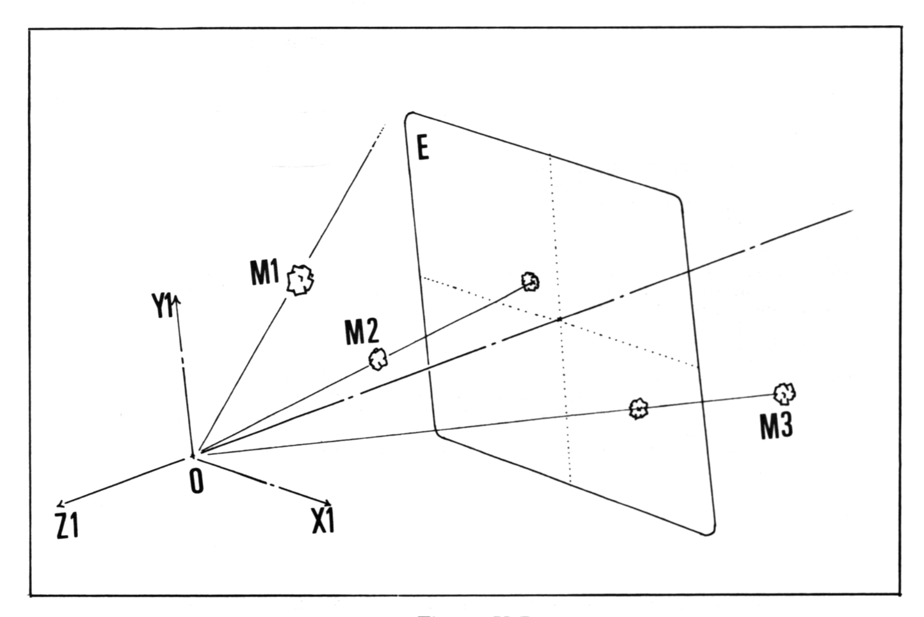


Figure X.B

Il en est de même ici. D'une façon générale, ces dispositifs optiques sont caractérisés par un angle qui définit la largeur de leur champ de vision. Cet angle peut être plus ou moins important (de 0 à 180° en hauteur, et de 0 à 360° horizontalement). L'œil humain ayant un champ de vision pas très large, on peut ainsi obtenir des effets amusants en utilisant sur un appareil photo un objectif "grand angle" *(fish-eye)*. Un basset ainsi photographié semble avoir une tête énorme et de toutes petites pattes. Il en est de même sur la Figure X.A, où le même polyèdre de forme oblongue (il ressemble à un crayon à papier) a été représenté selon un grand angle (environ 70°).

4. Exemple pratique : polyèdres

Cette figure a été obtenue à l'aide du Programme X.1, destiné à être amalgamé au programme *Tous polyèdres* (voir Chapitre VI).

```
PROGRAMME X.1

550 D=200
625 Prod=(D*c1*s2-e*a(t,0,0))*n(t,0)+(D*s1*s2-e*a(t,0,1))*n(t,1)+
(D*c2-e*a(t,0,2))*n(t,2)
1210 X1=(s1*x-c1*y)*e
1220 Y1=(s2*z-c1*c2*x-s1*c2*y)*e
1230 Z1=(c2*z+c1*s2*x+s1*s2*y)*e-D
1240 IF Z1>-1.125 GOTO 1500
1250 XX=-X1*115/Z1
1260 YY=-Y1*115/Z1
1270 RETURN
1500 '===================================================== Points lointains
1510 IF X1=0 AND Y1=0 THEN IF Z1<0 THEN XX=0:YY=0:RETURN ELSE RETURN
1520 k=30000/SQR(X1*X1+Y1*Y1)
1530 XX=k*X1
1540 YY=k*Y1
1550 RETURN
2053 a=INKEY(0):IF a<>-1 THEN GOSUB 1900:D=D-b:GOTO 600
2057 a=INKEY(2):IF a<>-1 THEN GOSUB 1900:D=D+b:GOTO 600
2645 PRINT:PRINT"_ "+CHR$(240)+" ou "+CHR$(241)" Pour faire avancer ou
reculer""le Polyedre."
2825 PRINT:PRINT"_ "+CHR$(240)+" ou "+CHR$(241)" Pour faire avancer ou
reculer""le Polyedre."
```

Ce programme permet de tracer les polyèdres en perspective optique. Le dispositif est celui de la Figure X.B. L'écran est placé à une distance de 115 par rapport à O, ce qui donne un grand angle. En effet, cette distance de 115, agrandie ou diminuée, diminue ou agrandit l'angle de visée.

Le point O se trouve du même côté de l'écran que vous, ou plutôt que votre œil. Ainsi, vous aurez effectivement l'impression que votre œil se trouve en O. La position relative du polyèdre par rapport à l'écran est, comme pour les météores, totalement indifférente : il peut être devant comme M_2, ou derrière comme M_3. Souvenez-vous qu'il en était de même lors des projections classiques. L'écran est en fait un plan de vision fictif, le seul vrai étant celui de la rétine, ou la pellicule.

Le repère X_1, Y_1, Z_1 est le même que d'habitude, à un détail près : le nombre D, qui est la distance entre O et le polyèdre. D en effet ne saurait être nul : on ne verrait rien. D'où une légère différence dans le calcul de Z_1 (voir lignes 1210 à 1230).

Pour qu'un objet soit visible, il faut que les coordonnées projetées XX et YY ne soient pas trop grandes (pour ne pas sortir de l'écran), mais aussi il faut que Z_1 soit négatif. Dans le cas contraire, l'objet est de l'autre côté de O, derrière vous. Dans ce cas, le point est placé très loin, sur un cercle de rayon 30000 (XX et YY ne doivent pas dépasser 32760). S'il y a un trait à tracer, il sortira de l'écran, vers ce point à l'infini, ou presque (lignes 1500 et suivantes). Cela se fait également si Z_1 est négatif, mais faible en valeur absolue. Cela se comprend quand on voit le calcul de XX et YY (lignes 1530 et 1540) : si Z_1 est trop proche de zéro, XX et YY vont dépasser la valeur fatidique de 32760, et il y aura une erreur. D'où ce nombre de 1,125, égal (à peu près) à $320 \cdot 115/32760$. X_1 et Y_1 étant supposés ne pas dépasser la valeur 320.

Mais d'où vient donc cette formule ?

C'est assez simple. On a un point M de coordonnées (X_1, Y_1, Z_1). On cherche l'intersection de la droite passant par O $(0,0,0)$ et du plan d'équation $Z_1 = -115$. Ce point d'intersection m, de coordonnées (XX, YY, ZZ), est sur le plan : donc ZZ = 115. D'autre part, il est sur la droite OM. Donc (voir Chapitre II, *Équation paramétrique d'une droite*), il existe un certain k tel que $XX = X_1 \cdot k$, $YY = Y_1 \cdot k$, $ZZ = Z_1 \cdot k$. Vu la valeur de ZZ, $k = -115/Z_1$, et voilà XX et YY. C'est tout simple. Il suffit de regarder les éléments dont on dispose.

Un peu plus compliqué : comment savoir si une face est cachée ou non ? Là encore, réfléchissons. La face est cachée si le point O se trouve du

même côté de son plan que le reste du polyèdre. Or l'équation du plan d'une face qui passe par le point (a_0, a_1, a_2), et de vecteur normal (n_0, n_1, n_2), est :

$$(x - a_0) \cdot n_0 + (y - a_1) \cdot n_1 + (z - a_2) \cdot n_2 = 0$$

c'est-à-dire la nullité du produit scalaire MA par le vecteur normal. Lorsque ce même produit scalaire est positif, le point est à l'extérieur de la face, vu la manière dont le vecteur normal a été choisi.

D'autre part, le point O a pour coordonnées dans le repère x, y, z $(D \cdot c_1 \cdot s_2, D \cdot s_1 \cdot s_2, D \cdot c_2)$, car il se trouve sur l'axe OZ_1 (dirigé par le vecteur normal à l'écran habituel $(c_1 \cdot s_2, s_1 \cdot s_2, c_2)$), à la distance D.

Dès lors, il suffit de remplacer x, y, z par ces coordonnées dans le produit scalaire précédent, pour savoir si O se trouve du bon côté, et donc si la face est visible ou non. Tout cela pour des polyèdres convexes, cela va sans dire. On retrouve tout ce calcul dans la ligne 625, modifiée comme il faut.

A propos de cette distance D, sachez encore qu'elle est initialisée à 200 (moyen) (ligne 550), mais que vous pouvez la faire augmenter ou diminuer avec les touches du curseur, pour donner l'illusion de vous rapprocher ou de vous éloigner.

Il ne vous reste plus qu'à essayer l'ultime programme de ce livre. Si vous trouvez la déformation un peu excessive (elle l'est, mais c'est pour qu'on la voie bien), augmentez simplement la valeur 115, et changez en même temps le 1,125 de la ligne 1240. Il faut mettre à la place votre nombre, multiplié par 320/32760.

Enfin, si le cœur vous en dit, rien ne vous empêche d'adapter ce type de programme sur des programmes plus complexes, comme ceux du Chapitre IX, ou encore le VII.1. Pour ce dernier, je ne l'ai pas fait, quoiqu'on puisse certainement obtenir des effets très intéressants (à condition de pouvoir aussi se déplacer vers la droite ou la gauche, et vers le haut ou le bas, ce qui n'est pas trop compliqué encore). En effet, le Programme VII.1 seul est assez lent, et cela ne pourrait que le ralentir un peu plus. D'autre part, la gestion des parties cachées est plus compliquée, car les verticales ne sont plus parallèles : on se retrouve alors, en fait, devant le même problème que pour les polyèdres concaves. J'avais déjà indiqué comment le résoudre à ce moment : cela n'a pas du tout changé. Simplement les calculs sont un peu plus longs.

Donc, là encore, je dois m'arrêter pour des raisons de lenteur. Si vous tenez absolument à venir à bout de tels problèmes, passez tout de suite au chapitre suivant, le tout dernier...

XI
POUR CONCLURE...

Afin de ne pas rester sur de mauvaises impressions, je vais développer dans ce court chapitre final quelques considérations générales comme au Chapitre VII, qui, je le souhaite, vous permettront peut-être d'ajouter à votre palette tridimensionnelle quelques éléments qui la rendront plus rapide, plus performante, ou simplement plus esthétique.

En effet, je rappelle que tous les programmes de ce livre ne sont pas réellement destinés à être utilisés tels quels, mais plutôt à servir d'exemples, pour que vous puissiez plus aisément construire les VÔTRES, le but global de cet ouvrage n'étant pas de limiter votre champ d'investigations, bien au contraire.

1. OPTIMISATION DES ANIMATIONS

L'accélération des effets d'animation, comme par exemple la rotation des objets, que nous avons vue, n'est pas réellement possible en BASIC. En effet, j'ignore si vous l'avez noté, mais l'ordre DRAW en particulier, que nous n'avons pratiquement cessé d'utiliser, est d'exécution fort longue, comparativement aux ordres de calculs.

Toutefois, certains effets intéressants peuvent être obtenus sans trop de mal. Par exemple, lors du tracé d'un polyèdre, on peut obtenir une sorte de "fondu enchaîné", en n'effaçant pas l'écran, mais en pratiquant ainsi : face par face, on commence par effacer l'ancienne face (en retraçant dessus en noir), puis l'on trace la nouvelle. Ainsi le polyèdre ne semble pas quitter l'écran, mais à la fin du tracé sa position a changé.

Autre possibilité : au lieu, par exemple, de faire une rotation de dix degrés, faites dix rotations de un degré à la chaîne. Cela sera plus long évidemment, mais l'effet d'animation sera meilleur. Nous verrons d'ailleurs au prochain paragraphe comment rendre les calculs plus rapides pour de tels effets.

Encore un exemple. Si vous avez un personnage qui vous représente dans un jeu, qui tourne sur lui-même dans un paysage de surface en trois dimensions, plutôt que de faire retracer toute la surface faites-lui simplement faire un décalage, vers la droite par exemple, chaque colonne remplaçant celle qui était à sa droite ; la colonne la plus à droite sera stockée en mémoire, la plus à gauche viendra de la mémoire. Naturellement, cela n'est guère possible qu'en langage machine, mais si votre programme est en BASIC, rien ne vous empêche de créer une instruction supplémentaire à cet effet.

Naturellement je ne doute pas que bien d'autres effets sont encore possibles ; il suffit que vous trouviez ceux qui sont exactement adaptés à votre cas...

2. OPTIMISATION DES CALCULS

Il existe de nombreuses manières d'optimiser les calculs. La plus importante est assez simple, mais exige une attention soutenue ; elle consiste à utiliser le bon type de variables, et en particulier à toujours

définir comme entières les variables qui le sont, et ce, même dans les boucles FOR...NEXT. Cela paraît facile, mais en réalité il faut faire très attention ; l'erreur classique en la matière consiste à faire : DEFINT *a*, parce que *a* est entier, mais sans penser que *aa* ne l'est pas.

Deuxième optimisation, que j'ai cherché à exploiter au maximum dans tous mes programmes : faire le minimum de calculs longs, comme les cosinus, sinus et autres tangentes et exponentielles. Ne pas recalculer cent fois le sinus de l'angle psi, si cet angle ne change pas. Le calcul de ces fonctions est en effet fort long, car basé sur des sommes de séries. Il faut donc l'éviter au maximum.

L'idéal serait également d'éviter les matrices, qui obligent la machine à chercher ses données en mémoire. Cela étant en général impossible, on peut toutefois éviter des ralentissements dans la recherche de ces données, en évitant de surcharger les programmes par des détails inutiles. Ainsi les lignes de REM qui peuplent mes programmes et sont fort longues servent uniquement à les rendre plus clairs. Mais une fois que vous les avez compris, rien ne vous oblige à les conserver, bien au contraire.

Au passage, je dois toutefois préciser que la présence de nombreux GOTO et GOSUB ne ralentit pratiquement pas les programmes. Cela est dû à une particularité du BASIC de l'Amstrad, qui remplace à la première exécution du programme tous ces ordres par les JUMP et CALL adéquats (aux adresses concernées). L'exécution du programme est ainsi plus rapide. En contrepartie, les listings sont plus lents. On ne peut pas tout avoir.

3. DES INCONVENIENTS DU BASIC : AUTRES LANGAGES UTILISABLES

Toutefois, ces petits trucs sont bien pauvres (s'il en existait de géniaux, je les aurais mis dans mes programmes), et de toute façon on se retrouve toujours bloqué par un obstacle fondamental : le BASIC. Ce langage a évidemment pour lui l'avantage de sa relative simplicité, mais en contrepartie il est fort lent.

Vous savez certainement qu'il existe de nombreux autres langages de programmation, dont un certain nombre sont déjà disponibles sur Amstrad.

Naturellement, tous ne sont pas adaptés au graphisme en trois dimensions. En particulier, il faut exclure les langages de gestion. Par contre, sont tout à fait utilisables le Pascal, le Fortran, même à la rigueur le Logo... liste non exhaustive évidemment.

Cependant, il ne faut pas perdre de vue une chose qui est essentielle : le microprocesseur, lui, travaille de toute façon en langage machine, quel que soit le langage que vous utilisez en programmation. Les langages cités plus haut, à commencer par le BASIC, sont dits interprétés, car ils nécessitent un *interpréteur* qui traduit le langage de programmation en langage machine.

Le seul langage non interprété est donc le langage machine lui-même, ou assembleur. Ce langage (il n'y a pas de réelle différence entre l'assembleur et le langage machine) est assez difficile (surtout sur le Z80, le microprocesseur des Amstrad), mais en revanche il est d'une rapidité défiant toute concurrence : en moyenne vingt fois plus grande que le BASIC.

Vous avez le choix entre utiliser un compilateur qui traduira vos programmes BASIC en langage machine (en vente dans le commerce), ou, ce qui est plus riche mais plus dur (encore une fois, on n'a rien sans rien), programmer directement en assembleur : les voies qui vous seront ainsi offertes sont infinies, à condition d'avoir la force de s'y engager...

Cependant, même si vous ne parvenez pas à faire de gros programmes en assembleur, même de petits peuvent vous servir. Ainsi vous trouverez, dans des ouvrages sur le langage machine, comment accélérer le calcul d'une racine carrée, par exemple (votre Amstrad calcule bêtement $\exp(\log(x)/2)$). Vous pourrez faire toutes sortes d'extensions d'instructions (j'en ai cité une au Paragraphe 1) : je vous laisse le soin de les imaginer, car les possibilités de l'assembleur sont telles qu'elles vous viendront spontanément à l'esprit.

Notez en particulier que tout ce qui est graphique est assez facile à programmer en assembleur, et immensément plus rapide qu'en BASIC.

4. ET DANS L'AVENIR ?

Ces spéculations sur l'assembleur ne sont pas seulement des vœux pieux ou des indications légères. Souvenez-vous en effet que j'ai renoncé à vous

donner deux programmes en BASIC : les polyèdres concaves et l'appartement vu en perspective optique, pour cause de lenteur !

Aussi, si je parviens à les mettre au point, j'espère avoir l'occasion de les publier séparément, programmés en assembleur ou en Pascal.

Enfin, vous aurez peut-être noté des absences d'envergure dans cet ouvrage : il s'agit des fractales. En effet, les possibilités offertes par ces objets mathématiques encore récents (ils datent de 1970 environ) sont si importantes, que je souhaite vivement avoir un jour la possibilité de publier un autre livre qui leur serait intégralement consacré.

En attendant, le présent ouvrage est terminé. S'il vous a permis, ne serait-ce qu'une fois, de programmer par vous-même en trois dimensions, en vous échappant du cadre rigoureux de mes programmes, alors il aura intégralement rempli les buts qu'il s'était fixé. Je le souhaite de tout cœur.

ANNEXE A
THEORIE ET PRATIQUE RESUMEES DE LA DERIVATION

J'ai déjà eu l'occasion dans le texte de parler de courbes, de surfaces, et de tangentes ou plans tangents. Comment les calculer, comment les utiliser, voilà ce qu'explique la théorie de la dérivation (j'ai déjà aussi parlé de dérivées). Cette théorie, qui est fondamentale en mathématiques, n'est pas réellement très compliquée (elle est enseignée essentiellement en classe de première), mais je l'ai mise en annexe, car elle n'est pas absolument nécessaire à la compréhension de la majeure partie de cet ouvrage.

1. NOTION DE TANGENTE A UNE COURBE

Sur la Figure A.A, en haut, on voit une courbe, et sur cette courbe, on a marqué, arbitrairement, un point M. Un peu plus loin sur la courbe, on pose un point H que l'on va faire se déplacer imaginairement sur la courbe, de sorte qu'il se rapproche petit à petit de M (en suivant la flèche en pointillé) ; trois positions du point H ont été marquées, mais j'aurais pu en mettre d'autres plus proches de M, si je n'avais craint de rendre la figure illisible. Or, pendant que H s'approche de M, observez la droite MH. Cette droite se redresse tout doucement. Finalement, lorsque H arrive tout près de M, la droite HM est pratiquement confondue avec la droite T marquée sur la figure.

Cette droite porte le nom de *tangente* à la courbe au point M. On voit assez bien en fait ce qu'est une tangente : c'est une droite qui ne fait qu'effleurer en quelque sorte la courbe, exactement au point M.

La tangente à une courbe en un point est distinguée par une double flèche de chaque côté du point, comme pour le point A. Pourquoi une double flèche, et non une simple ? En observant l'autre côté de la courbe par rapport à M, vous remarquerez que si le point H avait approché M en arrivant de l'autre côté (par en bas), la droite HM serait quand même venue se confondre avec T.

Cela n'est pas le cas forcément sur tous les points d'une courbe. Ainsi en B, voyez que l'on a porté ce qu'on appelle deux demi-tangentes (l'expression est claire, je pense). Ces deux demi-tangentes ne sont pas parallèles ; on dit que B est un point *anguleux*. Ce n'est pas très fréquent, mais cela arrive parfois sur certaines fonctions, comme la valeur absolue ($y = \text{abs}(x)$ est anguleux en zéro).

Ne confondez pas le point anguleux B avec un point d'inflexion comme C. Sur ce dernier, la courbe s'infléchit (d'où son nom) et traverse sa tangente (alors qu'en M, la courbe reste en dessous de la tangente), mais la tangente n'en est pas moins la même des deux côtés. Les points d'inflexion sont beaucoup plus nombreux.

C'est aussi de cette manière qu'on détermine la convexité d'une courbe. Un morceau de courbe est dit convexe si, en tous ses points, la courbe est au-dessus de sa tangente (cas du fragment BC), concave si elle est en dessous (cas du morceau MAB). On passe du convexe au concave et inversement quand on arrive en un point d'inflexion comme C, ou parfois en un point anguleux comme B.

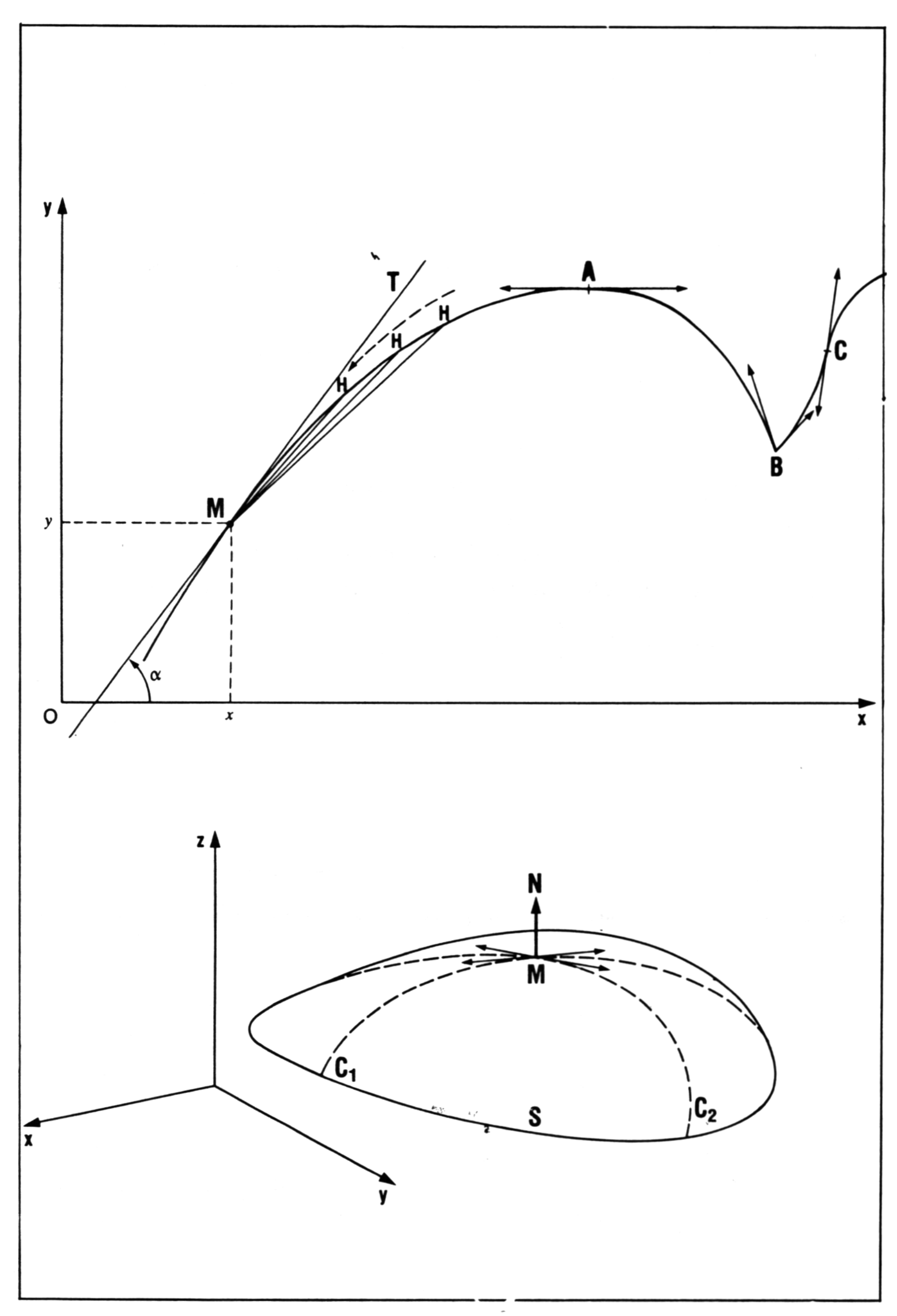


Figure A.A

D'autre part en A la tangente est horizontale (pente zéro). Notez que la fonction est alors fréquemment en un maximum ou un minimum de sa valeur.

2. EQUATION D'UNE TANGENTE — DERIVEE

On sait qu'une droite dans le plan a pour équation type : $y = a \cdot x + b$; a est appelée pente de la droite. C'est la tangente de l'angle qu'elle fait avec Ox (voir figure pour T). Dans le cas de T, si l'on connaît a, on peut trouver b facilement car l'on suppose connues les coordonnées (u, v) de M. En effet T passe par M, son équation est donc : $y - v = a \cdot (x - u)$ d'où $b = v - a \cdot u$. Reste donc à trouver a.

Si la courbe est du type $y = f(x)$ (f est une fonction ; on a donc $v = f(u)$), la valeur de a est appelée *dérivée de f au point u*. C'est une nouvelle fonction, que l'on note f' $\left(\text{ou encore } \frac{df}{dx}\right)$; en effet, la valeur de a est variable suivant le point où l'on se place (elle peut même ne pas exister, comme en B). On pourra donc trouver la tangente en tout point d'une courbe, à condition de trouver la fonction dérivée f'.

3. CALCUL D'UNE DERIVEE

Toute fonction f est égale à des sommes, produits, ..., composées (une composée est du type $f \circ g(x) = f[g(x)]$; par exemple, $\sin(\exp(x))$ est la composée de sin et exp : $\sin_o \exp$) de fonctions dites usuelles : puissances de x, exponentielles, fonctions trigonométriques, etc. Je vais donc donner à présent deux séries de formules : d'une part les dérivées des fonctions usuelles, d'autre part les dérivées de $f_o g$, $f \cdot g$, $f + g$ ..., f et g étant deux fonctions quelconques.

DÉRIVÉES DES FONCTIONS USUELLES

La dérivée est à droite, la fonction à gauche.

constante	0	x	1
x^k	$k \cdot x^{k-1}$	$1/x$	$-1/x^2$
sin (x)	cos (x)	cos (x)	$-$ sin (x)
tan (x)	$1 + \tan^2(x)$	atn (x)	$\frac{1}{(1+x^2)}$
sqr (x)	1/(2 sqr (x))	exp (x)	exp (x)
log (x)	$1/x$	$k \cdot x$	k

FORMULES DE DÉRIVATION

Idem. Les formules sont numérotées.

(1)	$f+g$	$f'+g'$
(2)	f^k	$k \cdot f' \cdot f^{k-1}$
(3)	$f \cdot g$	$f' \cdot g + g' \cdot f$
(4)	$1/f$	$-f'/f^2$
(5)	$\frac{f}{g}$	$\frac{(f' \cdot g - g' \cdot f)}{g^2}$
(6)	$f \circ g$	$g' \cdot f' \circ g$
(7)	$k \cdot f$	$k \cdot f'$

Dans toutes ces formules, k est une constante arbitraire. A partir de là, on peut trouver toutes les dérivées usuelles. Par exemple, cherchons ensemble la dérivée de la fonction $F(x) = \sin(2 \cdot x + 5)$. On peut écrire $F = f(g+h)$, avec $f = \sin$, $g(x) = 2 \cdot x$, $h(x) = 5$ (constante). Cherchons les dérivées de ces trois fonctions. Celle de f est $f' = \cos$; celle de g est $g' = 2$

(type $k \cdot x$) ; celle de h est $h' = 0$. Dès lors, posons $G = g + h$. On a alors $G' = g' + h' = 2$ (formule (1)). Puis $F = f(G) = f \circ G$. Donc :

$$F'(x) = G'(x) \cdot f' \circ G = 2 \cdot \cos(2 \cdot x + 5).$$

On a ainsi trouvé la dérivée de F. En procédant ainsi, on peut trouver une nouvelle formule :

(8) $f(k \cdot x + \ell)$ $\qquad$ $k \cdot f'(k \cdot x + \ell)$

Ainsi la dérivée de $\text{sqr}(k \cdot x + \ell)$ est $\dfrac{k}{(2 \cdot \text{sqr}\ (k \cdot x + \ell))}$, etc.

Calculons la dérivée de TAN sans utiliser la formule donnée, mais sachant que

$\tan(x) = \dfrac{\sin(x)}{\cos(x)} = \dfrac{f}{g}$ avec $f = \sin$ et $g = \cos$. On a $f' = \cos$ et $g' = -\sin$.

On utilise la formule (5) :

$$\tan'(x) = \frac{(\cos(x) \cdot \cos(x) - (-\sin(x)) \cdot \sin(x))}{\cos^2(x)}$$

$$= \frac{(\cos^2(x) + \sin^2(x))}{\cos^2(x)} = 1 + \left(\frac{\sin(x)}{\cos(x)}\right)^2 = 1 + \tan^2(x),$$

soit la valeur donnée. Notons aussi que, comme $\sin^2(x) + \cos^2(x) = 1$, cette dérivée est aussi égale à $1/\cos^2(x)$.

EXERCICE

Calculez de même la dérivée de $\text{cotan}(x)$. Vous devez obtenir

$$-(1 + \text{cotan}^2(x)) = \frac{-1}{\sin^2(x)}$$

Calculons à présent la dérivée de $\log(\exp(x)) = F(x)$. On a $F = f \circ g$, avec $f = \log$ $\left(\text{d'où } f'(x) = \dfrac{1}{x}\right)$ et $g = \exp$ (d'où $g' = \exp$; on dit que l'exponentielle est à elle-même sa propre dérivée). Donc par la formule (6) :

$$F' = \exp(x) \cdot \frac{1}{\exp(x)} = 1$$

ce qui n'est pas très surprenant car $\log(\exp(x)) = x$ en fait.

Calculons la dérivée de $\frac{1}{x^k} = F(x)$. On a $F = f \circ g$, avec $f(x) = 1/x$ $\left(\text{d'où } f'(x) = -1/x^2\right)$ et $g(x) = x^k$ (d'où $g'(x) = k \cdot x^{k-1}$). Dès lors par (6) :

$$F'(x) = k \cdot x^{k-1}\left(-\frac{1}{(x^k)^2}\right) = -\frac{k \cdot x^{k-1}}{x^{2k}} = -\frac{k}{x^{2k-(k-1)}} = -\frac{k}{x^{k+1}},$$

en simplifiant par x^{k-1} en haut et en bas de la fraction. Cette formule peut être obtenue très simplement en utilisant les puissances négatives : $x^{-k} = \frac{1}{x^k}$. Dès lors, en appliquant le calcul des dérivées usuelles en $-k$, on obtient :

$$(x^{-k})' = (-k) \cdot x^{-k-1} = -k \cdot x^{-(k+1)} = \frac{-k}{x^{k+1}}$$

soit effectivement F'.

Dernier exemple : calcul de la dérivée de $F = \frac{1}{\text{atn}(\text{sqr}(x))}$. On a

$$F = \frac{1}{H} \quad \text{avec} \quad H = f \circ g \quad \text{et} \quad f = \text{atn}, \quad g = \text{sqr}.$$

Donc $F' = -\frac{H'}{H^2}$ (formule (4)). Puis

$$H'(x) = g'(x) \cdot f'(g(x)) = \frac{1}{2 \cdot \text{sqr}(x)} \cdot \frac{1}{(1 + \text{sqr}^2(x))} =$$

$$\frac{1}{2 \quad \text{sqr}(x) \cdot (1 + x)}$$

(pour x positif, sinon $\text{sqr}(x)$ n'existe pas). Donc :

$$F'(x) = -1/(2 \cdot \text{sqr}(x) \cdot (1 + x) \cdot \text{atn}^2(\text{sqr}(x)))$$

EXERCICES

Démontrez les formules suivantes :

dérivée de $\exp(f) = f' \cdot \exp(f)$

dérivée de $\log(f) = f'/f$

Calculez les dérivées des fonctions suivantes :

1. $F(x) = \log(\sin(x))$
2. $F(x) = \exp(3 \cdot x^4 + 5 \cdot x - 7 + 2 \cdot \mathrm{sqr}(x))$
3. $F(x) = \sin^4(x)$
4. $F(x) = \cos(x)(2 + 3 \cdot \log(x - 1))$
5. $F(x) = \log(\mathrm{sqr}(x))$ (en déduire une expression plus simple de cette fonction. Montrez qu'elle résulte des formules : $\mathrm{sqr}(x) = x^k$ avec $k = \frac{1}{2}$, et $\log(x^k) = k \cdot \log(x)$).

Réponse 1 : $\tan(x)$

Réponse 2 : $(12\, x^3 + 5 + \frac{1}{\mathrm{sqr}\,(x)}) \cdot F(x)$

Réponse 3 : $4 \cdot \cos(x) \cdot \sin^3(x)$

Réponse 4 : $-\sin(x) \cdot (2 + 3 \cdot \log(x - 1)) + \frac{3 \cdot \cos(x)}{x - 1}$

Réponse 5 : $F'(x) = \frac{1}{2\,x}$ car $F(x) = \log(x)/2$.

4. DERIVEE PARTIELLE

Soit $F(x,y)$ une fonction de deux variables x et y. On appelle *dérivée partielle de F par rapport à x*, et l'on note F'_x, la dérivée de la fonction,

obtenue en considérant y comme une constante. Inversement, c'est x qu'on considère comme une constante pour le calcul de la dérivée partielle de F par rapport à y, F'_y. Par exemple :

$$F(x, y) = x + y \cdot \sin(x \cdot y)$$

Si l'on considère y comme une constante k, on a à dériver la fonction $G(x) = x + k \cdot \sin(k \cdot x)$. La dérivée de x est 1, celle de $\sin(k \cdot x)$ est $k \cdot \cos(k \cdot x)$, donc :

$$G'(x) = 1 + k \cdot (k \cdot \cos(k \cdot x) = 1 + k^2 \cdot \cos(k \cdot x).$$

Nous écrirons que $F'_x = 1 + y^2 \cdot \cos(x \cdot y)$.

Calculez à présent F'_y, en remplaçant x cette fois par k. Vous devez trouver $x \cdot \cos(x \cdot y) + \sin(x \cdot y)$.

5. UTILISATION

Regardez le bas de la Figure A.A. On y a représenté une surface type $z = F(x, y)$. En un point M, de coordonnées (a, b), il passe en particulier deux courbes planes tracées sur la surface : la courbe $z = f(x)$, avec pour tout x, $f(x) = F(x, b)$, et la courbe $z = g(y)$, avec $g(y) = F(a, y)$. Ces deux courbes sont appelées C_1 et C_2 sur la figure. Ces deux courbes admettent une tangente chacune en M. La pente de ces tangentes est $f'(a)$ pour C_1, et $g'(b)$ pour C_2. Voyez le paragraphe précédent : $f'(a) = F'_x(a, b)$ et $g'(b) = F'_y(a, b)$. Or, ces deux tangentes définissent un plan qui est appelé plan tangent à la surface en M. Vu ce qui a été dit sur les tangentes, il est aisé de trouver deux vecteurs dirigeant ce plan. En effet, une droite de pente a admet $(1, a)$, comme vecteur directeur. Dès lors le plan tangent à la surface en M peut être aisément trouvé : c'est le plan passant par M et dirigé par les deux vecteurs $(1, O, F'_x)$ et $(O, 1, F'_y)$.

Nous verrons comment tenir compte de tout cela dans l'Annexe B. C'est en effet surtout à la résolution d'équations implicites que les dérivées nous serviront, et au tracé de certaines courbes.

Je vous recommande donc d'essayer de bien comprendre cette annexe avant de passer à la suivante. Si vous trouvez tout cela vraiment trop compliqué, ne vous entêtez pas à l'excès: ce n'est pas absolument nécessaire.

ANNEXE B
EQUATIONS IMPLICITES RESOLUTION PRATIQUE D'EQUATIONS

Les éléments de cette annexe sont déjà utilisés dans certains programmes et chapitres de ce livre (notamment Chapitres VII et IX). Mais ils peuvent servir pour d'innombrables autres applications que je ne pourrais détailler.

1. RESOLUTION D'UNE EQUATION IMPLICITE : TRACE DE SA COURBE REPRESENTATIVE

Soit $F(x,y)=0$ une équation, dite implicite car sa solution n'est pas clairement exprimée, comme pour $y=f(x)$.

A une telle équation correspond une certaine courbe C en général dans le plan ; on dit que C est le lieu des points (x,y) satisfaisant à l'équation.

Admettons que nous souhaitions tracer cette courbe C. Comment faire ?

Il faut tout d'abord connaître au moins un point de C. Pour trouver ce point de départ, il faut poser par exemple $y=0$, et résoudre l'équation en x ainsi obtenue (voir deuxième partie de cette annexe). S'il n'y a pas de solution, on prend une autre valeur de y, jusqu'à ce qu'on finisse par trouver un premier point.

Une fois ce premier point trouvé, on va en chercher d'autres par une méthode approximative. Cette méthode est basée sur une remarque qu'il est facile de faire sur la Figure A.A : la courbe, au voisinage du point M, est pratiquement confondue avec sa tangente T ; il est difficile de les distinguer, à condition qu'on soit tout près de M, cependant.

Donnons-nous un nombre h très petit. Au lieu de chercher un point sur la courbe, situé à la distance h de M (de coordonnées x, y), nous allons prendre un point situé sur la *tangente* à la courbe en M, à la distance h. Ce point M′ ainsi obtenu n'est pas exactement sur la courbe, mais il y est presque : le tracé n'est donc en fait qu'approximatif, mais si h est assez petit, cela sera suffisant.

Reste à savoir quelle est la pente de la tangente à la courbe, afin de trouver les coordonnées de M′.

On dispose pour cela d'un théorème très important disant que la tangente en un point à la courbe a pour vecteur normal le vecteur de coordonnées (F'_x, F'_y). Cela est également vrai pour la surface $F(x,y,z)=0$, qui a pour vecteur normal (F'_x, F'_y, F'_z).

Dès lors, un vecteur directeur de cette tangente est, par exemple, $(F'_y, -F'_x)$. On peut donc poser pour coordonnées de M′ : $x+k\cdot F'_y$, $y-k\cdot F'_x$, avec $k=\dfrac{h}{r}$ et $r=\text{sqr}(F'^2_x+F'^2_y)$, pour que la distance entre M et M′ soit égale à h. Appelant alors à nouveau x et y ces deux nouvelles coordonnées, il n'y a plus qu'à recommencer et, de proche en proche, la courbe sera trouvée.

2. EXEMPLE D'APPLICATION : SURFACE F (x, y ,z) = 0

Admettons que nous voulions tracer une surface d'équation F $(x, y, z) = 0$, et que nous en connaissions les dérivées partielles de F. Deux méthodes sont possibles : celle du Chapitre IX, et celle du Chapitre VII.

Celle du Chapitre IX est (comme toujours) la plus simple. Il suffit de tracer les courbes à y constant, puis à x constant, ces deux nombres variant dans un domaine donné (voir Chapitre IX). Comment tracer de telles courbes ?

S'il s'agit par exemple de courbes à x constant, x valant donc x_0, l'équation F $(x_0, y, z) = 0$ est équivalente d'une équation du type G $(y, z) = 0$, équation implicite à deux variables. Il suffit donc de tracer la courbe correspondante, en augmentant y et z petit à petit. Ce travail est d'autant plus aisé que $G'_y(y, z) = F'_y(x_0, y, z)$ et idem pour les dérivées partielles par rapport à z. On se trouve donc ramené au cas précédent d'une simple courbe à équation implicite.

Pour faire comme au Chapitre VII, c'est-à-dire tracer l'horizon de la surface, c'est un peu plus compliqué.

Dans ce chapitre, j'avais expliqué que le vecteur normal (qui a pour coordonnées les trois dérivées partielles) permet de tracer cet horizon, car on peut ainsi en trouver l'équation, donnée par un produit scalaire du type

$$v_0 \cdot F'_x + v_1 \cdot F'_y + v_2 \cdot F'_z = 0$$

v_0, etc. étant les coordonnées de la perpendiculaire à l'écran.

Donc, posons $G = v_0 \cdot F'_x + \ldots$ Comme les trois dérivées partielles, G est une fonction de (x, y, z). C'est donc aussi l'équation d'une surface (différente, en principe). L'horizon est l'intersection de cette surface avec F $(x, y, z) = 0$ (car n'oublions tout de même pas que les points de l'horizon sont sur la surface de départ). Cet horizon a donc une double équation : F $(x, y, z) = 0$ et G $(x, y, z) = 0$. Cela définit en général une certaine courbe (pensez aux droites : si F et G sont des équations de plan, on obtient ainsi les équations intrinsèques de leur intersection).

Cela ne nous dit pas encore comment tracer une telle courbe.

C'est tout à fait possible, à condition de connaître un point de départ, comme d'habitude. Ensuite, que faire ? Réponse dans le paragraphe

précédent : trouver la tangente en un point M à la courbe, puis chercher sur cette tangente un point à la distance h, et recommencer.

Ainsi, si **t** est le vecteur tangent à la courbe en M, on posera :

$$x = x + \frac{h \cdot t_0}{r} \quad : \quad y = y + \frac{h \cdot t_1}{r} \quad : \quad z = z + \frac{h \cdot t_2}{r}$$

et avec évidemment t_0, t_1, t_2 les coordonnées de **t**, et r sa norme :

$$r = \mathrm{sqr}\,(t_0^2 + t_1^2 + t_2^2).$$

Reste donc seulement à trouver les coordonnées de **t**. Réfléchissons un peu : **t** est la direction de la tangente à l'horizon H. Or, cet horizon est tracé sur la surface $F(x, y, z) = 0$, comme C_1 sur S dans le bas de la Figure A.A. Sa tangente est donc dans le plan tangent en M à la surface : sa direction **t** est donc perpendiculaire au vecteur normal à la surface en M, dont les coordonnées sont (F'_x, F'_y, F'_z). De même **t** est perpendiculaire au vecteur normal à $G(x, y, z) = 0$, soit (G'_x, G'_y, G'_z). Dès lors, il suffit de prendre pour **t** le produit vectoriel de ces deux vecteurs normaux, et donc :

$$t_0 = F'_y \cdot G'_z - F'_z \cdot G'_y,$$

et ainsi de suite.

On peut ainsi tracer H. Mais c'est plus difficile, car il faut calculer non seulement les dérivées partielles de F, mais aussi celles de G. L'avantage des polynômes, c'est que l'ordinateur peut le faire sans trop de difficultés, seul, d'où la nature du Programme VII.2. Mais rien ne vous empêche de le faire vous-même, et de rentrer ces dérivées partielles comme des données, en même temps que F.

Notez d'autre part que si G est l'équation d'un plan, on obtient par la même méthode le tracé de l'intersection du plan avec la surface.

3. RESOLUTION D'EQUATION A UNE SEULE VARIABLE

Tout ce que nous avons trouvé au paragraphe précédent est fort intéressant, mais inutile si l'on ne trouve pas le point de départ dont nous avons parlé plusieurs fois.

Pour trouver ce point, on pose $y = y_0$, $z = z_0$, deux constantes choisies plus ou moins au hasard, et on espère que, en posant $f(x) = F(x, y_0, z_0)$, l'équation $f(x) = 0$ admettra au moins une solution (on dit aussi un zéro, ou une racine) x_0, qui nous donnera le point de départ (sinon, il faut changer les deux constantes et recommencer). Cette solution existera si la droite $y = y_0 : z = z_0$, parallèle à Ox, coupe la surface au moins en un point.

Il s'agit donc de résoudre l'équation $f(x) = 0$.

La valeur de x pouvant varier de l'infini négatif à l'infini positif, il faut chercher x seulement dans un domaine donné : x compris entre a et b.

De deux choses l'une : soit vous avez déjà une idée approximative de la valeur de x_0, et vous pouvez trouver a et b, soit vous ne savez rien. Dans ce cas, l'ordinateur peut (parfois) trouver lui-même a et b.

En effet, en général lorsque $f(x_0) = 0$, f est d'un certain signe avant x_0 (par exemple f est positif quand x est un peu inférieur à x_0), et du signe opposé (donc négatif dans ce cas), un peu après.

Inversement, si l'on peut trouver deux nombres a et b tels que $f(a)$ et $f(b)$ soient de signes contraires (ce qui se traduit par $f(a) \cdot f(b)$ négatif), on est sûr (sauf fonction très bizarre) qu'il a une racine entre a et b.

Donc, pour trouver a et b quand on ne sait rien, on procède ainsi : on appelle $k = f(0)$. Ensuite on teste $f(n)$ et $f(-n)$, en vérifiant pour chacun s'ils ont le même signe que k, en faisant varier n de 1 à l'infini. En fait, on s'arrête quand on a trouvé un nombre N, tel que $k \cdot f(N)$ soit négatif (on pose alors $a = N - 1$, $b = N$) ou tel que $k \cdot f(-N)$ soit négatif (on pose alors $a = -N$ *et* $b = -N + 1$).

Les deux bornes a et b étant trouvées, on procède ainsi : on pose $c = \dfrac{(a+b)}{2}$, le milieu de a et b, et on calcule $f(c)$. Si $f(c)$ est pratiquement nul (à une approximation donnée près, par exemple 0,1 près), alors $x_0 = c$. Sinon, si $f(c) \cdot f(a)$ est négatif, on pose $b = c$, et s'il est positif, on pose $a = c$. On a alors une nouvelle valeur du couple (a, b) et on recommence au début de l'algorithme.

En principe, la racine est trouvée au bout d'un certain nombre d'itérations.

Notons qu'il existe d'autres façon de calculer c. Par exemple, prendre pour c l'intersection de la droite ab avec l'axe des x, soit

$$c = \frac{a - f(a) \cdot (b - a)}{(f(b) - f(a))}$$

et bien d'autres encore, qui permettent selon les cas d'aboutir plus vite, ou de trouver même des racines difficiles (cas où f s'annule sans changer de signe, comme par exemple $f(x) = x^2$), mais au prix d'une plus grande complexité.

Le détail de ces méthodes sort du cadre de cet ouvrage. De toute façon, il n'existe aucune méthode absolument parfaite. Celle que j'ai donnée a un avantage important : sa simplicité. Elle vous permettra de trouver vos points de départ dans 99 pour 100 des cas, ce qui est, à mon sens, amplement suffisant.

4. EXEMPLE D'APPLICATION : DESSIN SUR UNE SURFACE

Soit une surface d'équation $F(x, y, z) = 0$ que nous appellerons S très originalement, et un dessin plan formé d'une succession de points (reliés entre eux ou non), obtenus à l'aide du plot baladeur par exemple. Ce dessin représente, supposons, le mot "Surface". Vous désirez reproduire ce dessin sur la surface, comme on grave sur une pomme ou un tronc. Comment faire ?

C'est assez simple. Choisissez d'abord un point M_0 sur la surface, autour duquel vous désirez que le dessin soit fait. Puis prenez un point A à l'intérieur de la surface (par exemple le centre si S est une sphère ou un ellipsoïde). Calculez le vecteur normal $\mathbf{N_0}$ à S en M_0.

Soit u_0, v_0, w_0 ses coordonnées. Il y correspond deux angles en sphériques. Vous pouvez ainsi trouver un repère M_0, x_0, y_0, z_0, dans lequel le plan P tangent en M_0 à S est égal à x_0 M_0 y_0, par une rotation puis une translation. Soit (a, b, c) les coordonnées de A dans ce repère, et soit un point du dessin. Ce dessin est supposé, dans un premier temps, se trouver sur le plan P. Un point de coordonnées (X, Y) du dessin est donc placé en (X, Y, O) dans ce repère R_0. A présent cherchons le point (en principe unique) qui se trouve à l'intersection de S et de la droite joignant A à M (X, Y, O). Ce point P a pour coordonnées $X + k(X - a)$, $Y + k(Y - b)$, $-k \cdot c$, où k est un paramètre inconnu mais qui est tel que $F(X + k(X - a),\ Y + k(Y - b),\ -k \cdot c) = 0$, ce qui peut encore s'écrire $f(k) = 0$. Il suffit donc, pour trouver P, de résoudre cette équation. Si A a été correctement choisi (dans la concavité de la surface en fait), $a = 0$ et $b = 1$ conviennent comme bornes, car P est alors situé entre A et M.

On trouve ainsi P pour tous les M du dessin, et on le relie de la même manière ; le dessin se trouve alors reproduit sur la surface S. Victoire !

Bien d'autres exemples d'utilisation seraient encore possibles. D'une façon générale, tout ce qui fait intervenir des fonctions un peu compliquées utilise peu ou prou la résolution d'équations. C'est pourquoi sa maîtrise est extrêmement utile.

LA BIBLIOTHÈQUE SYBEX

OUVRAGES GÉNÉRAUX

VOTRE PREMIER ORDINATEUR *par Rodnay Zaks,*
296 pages, Réf. 394

VOTRE ORDINATEUR ET VOUS *par Rodnay Zaks,*
296 pages, Réf. 271

DU COMPOSANT AU SYSTÈME : une introduction aux microprocesseurs *par Rodnay Zaks,*
636 pages, Réf. 0040

TECHNIQUES D'INTERFACE aux microprocesseurs *par Austin Lesea et Rodnay Zaks,*
450 pages, Réf. 0039

LEXIQUE INTERNATIONAL MICRO-ORDINATEURS, avec dictionnaire abrégé en 10 langues
192 pages, Réf. 234

GUIDE DES MICRO-ORDINATEURS A MOINS 3 000 F *par Joël Poncet,*
144 pages, Réf. 322

LEXIQUE MICRO-INFORMATIQUE *par Pierre Le Beux,*
140 pages, Réf. 369

LA SOLUTION RS-232 *par Joe Campbell,*
208 pages, Réf. 0052

MINITEL ET MICRO-ORDINATEUR *par Pierrick Bourgault,*
198 pages, Réf. 0119

ROBOTS - CONSTRUCTION, PROGRAMMATION *par Fernand Esteves,*
400 pages, Réf. 0130

ALGORITHMES *par Pierre Beaufils et Wolfram Luther,*
296 pages, Réf. 0149

BASIC

VOTRE PREMIER PROGRAMME BASIC *par Rodnay Zaks,*
208 pages, Réf. 263

INTRODUCTION AU BASIC *par Pierre Le Beux,*
336 pages, Réf. 0035

LE BASIC PAR LA PRATIQUE : 60 exercices *par Jean-Pierre Lamoitier,*
252 pages, Réf. 0095

LE BASIC POUR L'ENTREPRISE *par Xuan Tung Bui,*
204 pages, Réf. 253

PROGRAMMES EN BASIC, Mathématiques, Statistiques, Informatique *par Alan R. Miller,*
318 pages, Réf. 259

BASIC, PROGRAMMATION STRUCTURÉE *par Richard Mateosian,*
352 pages, Réf. 0129

JEUX D'ORDINATEUR EN BASIC *par David H. Ahl,*
192 pages, Réf. 246

NOUVEAUX JEUX D'ORDINATEUR EN BASIC *par David H. Ahl,*
204 pages, Réf. 247

FICHIERS EN BASIC *par Alan Simpson,*
256 pages, Réf. 0102

TECHNIQUES DE PROGRAMMATION EN BASIC *par S. Crosmarie, M. Perron et D. Philippine*
152 pages, Réf. 0124

PASCAL

INTRODUCTION AU PASCAL *par Pierre Le Beux,*
496 pages, Réf. 0030

LE PASCAL PAR LA PRATIQUE *par Pierre Le Beux et Henri Tavernier,*
562 pages, Réf. 361

LE GUIDE DU PASCAL *par Jacques Tiberghien,*
504 pages, Réf. 423

PROGRAMMES EN PASCAL pour Scientifiques et Ingénieurs *par Alan R. Miller,*
392 pages, Réf. 240

AUTRES LANGAGES

INTRODUCTION A ADA *par Pierre Le Beux,*
366 pages, Réf. 360

INTRODUCTION A C *par Bruce Hunter,*
312 pages, Réf. 0092

MICRO-ORDINATEURS

ALICE

JEUX EN BASIC POUR ALICE *par Pierre Monsaut,*
96 pages, Réf. 320

ALICE et ALICE 90, PREMIERS PROGRAMMES *par Rodnay Zaks,*
248 pages, Réf. 376

ALICE, 56 PROGRAMMES *par Stanley R. Trost,*
160 pages, Réf. 401

ALICE, GUIDE DE L'UTILISATEUR *par Norbert Rimoux,*
208 pages, Réf. 378

ALICE, PROGRAMMATION EN ASSEMBLEUR *par Georges Fagot-Barraly,*
192 pages, Réf. 420

AMSTRAD

AMSTRAD, PREMIERS PROGRAMMES *par Rodnay Zaks,*
248 pages, Réf. 0105

AMSTRAD, 56 PROGRAMMES *par Stanley R. Trost,*
160 pages, Réf. 0107

AMSTRAD, JEUX D'ACTION *par Pierre Monsaut,*
96 pages, Réf. 0108

AMSTRAD, PROGRAMMATION EN ASSEMBLEUR
par Georges Fagot-Barraly,
208 pages, Réf. 0136

AMSTRAD EXPLORÉ *par John Braga,*
192 pages, Réf. 0135

AMSTRAD, GUIDE DU GRAPHISME *par James Wynford,*
208 pages, Réf. 0141

AMSTRAD CP/M 2.2 *par Anatole D'Hardencourt,*
248 pages, Réf. 0156

AMSTRAD ASTROLOGIE/NUMEROLOGIE/BIORYTHMES
par Pierrick Bourgault,
160 pages, Réf. 0167

AMSTRAD MULTIPLAN *de Microsoft,*
496 pages, Réf. 1111

AMSTRAD, CRÉER DE NOUVELLES INSTRUCTIONS
par Jean-Claude Despoine,
144 pages, Réf. 0176

AMSTRAD ASTROCALC
par Gérard Blanc et Philippe Destrebecq,
168 pages, Réf. 0162

APPLE / MACINTOSH

PROGRAMMEZ EN BASIC SUR APPLE II,
Tomes 1 et 2 *par Léopold Laurent,*
208 pages, Réf. 333 et 380

APPLE II 66 PROGRAMMES BASIC *par Stanley R. Trost,*
192 pages, Réf. 283

JEUX EN PASCAL SUR APPLE
par Douglas Hergert et Joseph T. Kalash,
372 pages, Réf. 241

GUIDE DU BASIC APPLE II *par Douglas Hergert,*
272 pages, Réf. 0006

APPLE II, PREMIERS PROGRAMMES *par Rodnay Zaks,*
248 pages, Réf. 373

MACINTOSH, GUIDE DE L'UTILISATEUR
par Joseph Caggiano,
208 pages, Réf. 396

APPLE IIC, GUIDE DE L'UTILISATEUR
par Thomas Blackadar,
160 pages, Réf. 0089

MULTIPLAN SUR MACINTOSH
par Goulven Habasque,
240 pages, Réf. 0099

INTRODUCTION A MAC PASCAL *par Pierre Le Beux,*
416 pages, Réf. 0145

MACINTOSH POUR LA PRESSE, L'EDITION ET
LA PUBLICITE *par Bernard Le Du,*
160 pages, Réf. 0173

ATARI

JEUX EN BASIC SUR ATARI *par Paul Bunn,*
96 pages, Réf. 282

ATARI, PREMIERS PROGRAMMES *par Rodnay Zaks,*
248 pages, Réf. 387

ATARI, GUIDE DE L'UTILISATEUR *par Thomas Blackadar,*
192 pages, Réf. 354

ATMOS

JEUX EN BASIC SUR ATMOS *par Pierre Monsaut,*
96 pages, Réf. 346

ATMOS, 56 PROGRAMMES *par Stanley R. Trost,*
180 pages, Réf. 372

COMMODORE 64

JEUX EN BASIC SUR COMMODORE 64
par Pierre Monsaut,
96 pages, Réf. 0017

COMMODORE 64, PREMIERS PROGRAMMES
par Rodnay Zaks,
248 pages, Réf. 342

GUIDE DU BASIC VIC 20, COMMODORE 64
par Douglas Hergert,
240 pages, Réf. 312

COMMODORE 64, GUIDE DE L'UTILISATEUR
par J. Kascmer,
144 pages, Réf. 314

COMMODORE 64, 66 PROGRAMMES
par Stanley R. Trost,
192 pages, Réf. 319

COMMODORE 64, GUIDE DU GRAPHISME
par Charles Platt,
372 pages, Réf. 0053

COMMODORE 64, JEUX D'ACTION *par Eric Ravis,*
96 pages, Réf. 403

COMMODORE 64, 1ers CONTACTS
par Marty Dejonghe et Caroline Earhart,
208 pages, Réf. 390

COMMODORE 64, BASIC APPROFONDI
par Gary Lippman,
216 pages, Réf. 0100

DRAGON

JEUX EN BASIC SUR DRAGON *par Pierre Monsaut,*
96 pages, Réf. 324

EXL 100

EXL 100, JEUX D'ACTION *par Pierre Monsaut,*
96 pages, Réf. 0126

GOUPIL

PROGRAMMEZ VOS JEUX SUR GOUPIL
par François Abella,
208 pages, Réf. 264

HECTOR

HECTOR JEUX D'ACTION *par Pierre Monsaut,*
96 pages, Réf. 388

IBM

IBM PC EXERCICES EN BASIC *par Jean-Pierre Lamoitier,*
256 pages, Réf. 338

IBM PC GUIDE DE L'UTILISATEUR
par Joan Lasselle et Carol Ramsey,
160 pages, Réf. 301

IBM PC 66 PROGRAMMES BASIC *par Stanley R. Trost,*
192 pages, Réf. 359

GRAPHIQUES SUR IBM PC *par Nelson Ford,*
320 pages, Réf. 357

GUIDE DE PC DOS *par Richard A. King,*
240 pages, Réf. 0013

LASER

LASER JEUX D'ACTION *par Pierre Monsaut,*
96 pages, Réf. 371

MO 5

MO 5 JEUX D'ACTION *par Pierre Monsaut,*
96 pages, Réf. 0067

MO 5, PREMIERS PROGRAMMES *par Rodnay Zaks,*
248 pages, Réf. 370

MO 5, 56 PROGRAMMES *par Stanley R. Trost,*
160 pages, Réf. 375

MO 5, PROGRAMMATION EN ASSEMBLEUR
par Georges Fagot-Barraly,
192 pages, Réf. 384

MO 5, DYNAMIQUE CINÉMATIQUE, MÉTHODE POUR LA PROGRAMMATION DES JEUX *par Daniel Lebigre,*
272 pages, Réf. 0118

MO 5, STATIQUE, DYNAMIQUE, ELECTRONIQUE, PROGRAMMES DE PHYSIQUE EN BASIC
par Beaufils, Lamarche et Muggianu,
240 pages, Réf. 0148

MO 5, PROGRAMMES D'ELECTRONIQUE EN BASIC
par Beaufils, Delusurieux, Do, Romanacce,
312 pages, Réf. 0143

MO 5, OPTIQUE, THERMODYNAMIQUE, CHIMIE
par P. Beaufils, M. Lamarche, Y. Muggianu,
224 pages, Réf. 0161

MSX

MSX, JEUX D'ACTION *par Pierre Monsaut,*
96 pages, Réf. 411

MSX, INITIATION AU BASIC *par Rodnay Zaks,*
248 pages, Réf. 410

MSX, 56 PROGRAMMES *par Stanley R. Trost,*
160 pages, Réf. 0109

MSX, GUIDE DU GRAPHISME *par Mike Shaw,*
192 pages, Réf. 0132

MSX, PROGRAMMES EN LANGAGE MACHINE
par Steeve Webb,
112 pages, Réf. 0153

MSX, PROGRAMMATION EN ASSEMBLEUR
par Georges Fagot-Barraly,
216 pages, Réf. 0144

MSX, GUIDE DU BASIC *par Michel Laurent,*
264 pages, Réf. 0155

MSX, JEUX EN ASSEMBLEUR *par Eric Ravis*
112 pages, Réf. 0170

MSX, ROUTINES GRAPHIQUES EN ASSEMBLEUR
par Steeve Webb
88 pages, Réf. 0154

MSX, TECHNIQUES DE PROGRAMMATION DES JEUX EN ASSEMBLEUR
par Georges Fagot-Barraly,
176 pages, Réf. 0178

MSX ASTROLOGIE/NUMEROLOGIE/BIORYTHMES
par Pierrick Bourgault,
157 pages, Réf. 0168

ORIC

JEUX EN BASIC SUR ORIC *par Peter Shaw,*
96 pages, Réf. 278

ORIC PREMIERS PROGRAMMES *par Rodnay Zaks,*
248 pages, Réf. 344

SHARP

DÉCOUVREZ LE SHARP PC-1500 ET LE TRS-80 PC-2
par Michel Lhoir,
2 tomes, Réf. 261-262

SPECTRAVIDEO

SPECTRAVIDEO, JEUX D'ACTION *par Pierre Monsaut,*
96 pages, Réf. 377

SPECTRUM

PROGRAMMEZ EN BASIC SUR SPECTRUM
par S.M. Gee,
208 pages, Réf. 252

JEUX EN BASIC SUR SPECTRUM *par Peter Shaw,*
96 pages, Réf. 276

SPECTRUM, PREMIERS PROGRAMMES *par Rodnay Zaks,*
248 pages, Réf. 381

SPECTRUM JEUX D'ACTION *par Pierre Monsaut,*
96 pages, Réf. 368

TI 99/4

PROGRAMMEZ VOS JEUX SUR TI 99/4
par François Abella,
160 pages, Réf. 303

TO 7

JEUX EN BASIC SUR TO 7 *par Pierre Monsaut,*
96 pages, Réf. 0026

TO 7, PREMIERS PROGRAMMES *par Rodnay Zaks,*
248 pages, Réf. 328

TO 7, PROGRAMMATION EN ASSEMBLEUR
par Georges Fagot-Barraly,
192 pages, Réf. 350

JEUX SUR TO 7 et MO 5 *par Georges Fagot-Barraly,*
168 pages, Réf. 0134

GESTION DE FICHIERS SUR TO 7 ET MO 5
par Jean-Pierre Lhoir,
136 pages, Réf. 0127

TO 7, 56 PROGRAMMES *par Stanley R. Trost,*
160 pages, Réf. 374

TO 7 / MO 5, GUIDE DU BASIC
par Jean-Louis Greco et Michel Laurent,
288 pages, Réf. 0158

TO 7 / MO 5, GUIDE DU GRAPHISME
par Michel Lamarche et Yves Muggianu,
240 pages, Réf. 0172

TO 7 / MO 5 ASTROLOGIE/NUMEROLOGIE/BIORYTHMES
par Pierrick Bourgault,
160 pages, Réf. 0169

TRS-80

PROGRAMMEZ EN BASIC SUR TRS-80
par Léopold Laurent,
2 tomes, Réf. 366-251

JEUX EN BASIC SUR TRS-80 MC-10 *par Pierre Monsaut,*
96 pages, Réf. 323

JEUX EN BASIC SUR TRS-80 *par Chris Palmer,*
96 pages, Réf. 302

JEUX EN BASIC SUR TRS-80 COULEUR
par Pierre Monsaut,
96 pages, Réf. 325

TRS-80 MODÈLE 100, GUIDE DE L'UTILISATEUR
par Orson Kellog,
112 pages, Réf. 300

TRS-80 COULEUR, PREMIERS PROGRAMMES
par Rodnay Zaks,
248 pages, Réf. 414

TRS-80 COULEUR, 56 PROGRAMMES
par Stanley R. Trost,
160 pages, Réf. 413

VIC 20

PROGRAMMEZ EN BASIC SUR VIC 20
par G. O. Hamann,
2 tomes, Réf. 329-337

JEUX EN BASIC SUR VIC 20 *par Alastair Gourlay,*
96 pages, Réf. 277

VIC 20, PREMIERS PROGRAMMES *par Rodnay Zaks,*
248 pages, Réf. 341

VIC 20 JEUX D'ACTION *par Pierre Monsaut,*
96 pages, Réf. 345

VG 5000

VG 5000, JEUX D'ACTION *par Pierre Monsaut,*
96 pages, Réf. 422

VG 5000, 56 PROGRAMMES *par Stanley R. Trost,*
160 pages, Réf. 0128

ZX 81

ZX 81 GUIDE DE L'UTILISATEUR *par Douglas Hergert,*
208 pages, Réf. 351

ZX 81 56 PROGRAMMES BASIC *par Stanley R. Trost,*
192 pages, Réf. 304

GUIDE DU BASIC ZX 81 *par Douglas Hergert,*
204 pages, Réf. 285

JEUX EN BASIC SUR ZX 81 *par Mark Charlton,*
96 pages, Réf. 275

ZX 81 PREMIERS PROGRAMMES *par Rodnay Zaks,*
248 pages, Réf. 343

MICROPROCESSEURS

PROGRAMMATION DU Z80 *par Rodnay Zaks,*
618 pages, Réf. 0058

APPLICATIONS DU Z80 *par James W. Coffron,*
304 pages, Réf. 0181

PROGRAMMATION DU 6502 *par Rodnay Zaks,*
376 pages, Réf. 0031, 2ème édition

APPLICATIONS DU 6502 *par Rodnay Zaks,*
288 pages, Réf. 332

PROGRAMMATION DU 6800
par Daniel-Jean David et Rodnay Zaks,
374 pages, Réf. 327

PROGRAMMATION DU 6809
par Rodnay Zaks et William Labiak,
392 pages, Réf. 0139

PROGRAMMATION DU 8086/8088
par James W. Coffron,
304 pages, Réf. 0016

MISE EN OEUVRE DU 68000 *par C. Vieillefond,*
352 pages, Réf. 0133

ASSEMBLEUR DU 8086/8088 *par François Retoreau,*
616 pages, Réf. 0093

SYSTÈMES D'EXPLOITATION

GUIDE DU CP/M AVEC MP/M *par Rodnay Zaks,*
354 pages, Réf. 336

CP/M APPROFONDI *par Alan R. Miller,*
380 pages, Réf. 334

INTRODUCTION AU p-SYSTEM UCSD
par Charles W. Grant et Jon Butah,
308 pages, Réf. 365

GUIDE DE MS-DOS *par Richard A. King,*
360 pages, Réf. 0117

INTRODUCTION A UNIX *par John D. Halamka,*
240 pages, Réf. 0098

GUIDE DE PRODOS
par Pierre Beaufils et Wolfram Luther,
248 pages, Réf. 0146

APPLICATIONS ET LOGICIELS

INTRODUCTION AU TRAITEMENT DE TEXTE
par Hal Glatzer,
228 pages, Réf. 243

INTRODUCTION A WORDSTAR *par Arthur Naiman,*
200 pages, Réf. 0062

WORDSTAR APPLICATIONS *par* *Julie Anne Arca,*
320 pages, Réf. 0005

VISICALC APPLICATIONS *par* *Stanley R. Trost,*
304 pages, Réf. 258

VISICALC POUR L'ENTREPRISE *par* *Dominique Helle,*
304 pages, Réf. 309

INTRODUCTION A dBASE II *par* *Alan Simpson,*
280 pages, Réf. 0064

DE VISICALC A VISI ON *par* *Jacques Bourdeu,*
256 pages, Réf. 321

MULTIPLAN POUR L'ENTREPRISE
par D. Helle et G. Boussand,
304 pages, Réf. 0079

dBASE II APPLICATIONS *par* *Christophe Stehly,*
248 pages, Réf. 416

INTRODUCTION A LOTUS 1-2-3
par Chris Gilbert et Laurie Williams,
272 pages, Réf. 0106

INTRODUCTION A dBASE III *par* *Alan Simpson,*
272 pages, Réf. 0131

LOTUS 1-2-3 POUR L'ENTREPRISE
par Dominique Helle et Guy Boussand,
256 pages, Réf. 0147

LOTUS 1-2-3 PROGRAMMATION DES MACRO-COMMANDES *par* *Goulven Habasque,*
144 pages, Réf. 0150 F

LOGISTAT, ANALYSE STATISTIQUE DES DONNÉES
par Fredj Tekaia et Michele Bidel,
352 pages, Réf. 0115

ALGORITHMES *par* *P. Beaufils, et W. Luther,*
296 pages, Réf. 0149

POUR UN CATALOGUE COMPLET DE NOS PUBLICATIONS

FRANCE
6-8, Impasse du Curé
75881 PARIS CEDEX 18
Tél. : (1) 42.03.95.95
Télex : 211801

U.S.A.
2344 Sixth Street
Berkeley, CA 94710
Tel. : (415) 848.8233
Telex : 336311

ALLEMAGNE
Vogelsanger. WEG 111
4000 Düsseldorf 30
Postfach N° 30.09.61
Tel. : (0211) 61 80 2-0
Telex : 08588163



Paris • Berkeley • Düsseldorf

Achevé d'imprimer le 10 février 1986 sur les presses de l'Imprimerie «La Source d'Or»
63200 Marsat - Dépôt légal : 1er trimestre 1986 - Imprimeur n° 1955

Représenter des polyèdres, des surfaces, voire des objets plus complexes en trois dimensions sur l'écran de votre ordinateur. Les faire pivoter pour les observer sous tous leurs angles. Supprimer les parties qui, dans la réalité, seraient dissimulées au regard, afin de rendre l'impression de relief encore plus saisissante. Dessiner des lettres ou des petites images sur des plans qui semblent inclinés, sur des cônes, des sphères ou des cylindres...
Ce genre de réalisation ne nécessitera jamais plus de 200 à 300 lignes de BASIC et même souvent beaucoup moins sur votre Amstrad. Ces programmes d'une technique très particulière seront transposables assez facilement sur d'autres appareils.
Ce sont donc ces techniques de programmation que cet ouvrage se propose de vous faire découvrir. Les listings détaillés des programmes sont accompagnés de nombreuses figures et organigrammes. Le lecteur pourra adapter les programmes de ce livre au gré de sa fantaisie.
Laissez-vous donc entraîner au fil des pages, et vous parviendrez presque sans vous en apercevoir, à promener votre écran comme une caméra dans un labyrinthe diabolique, ou alors dans le salon d'une demeure imaginaire que l'ordinateur aura reconstitué pour vous !
Et la vocation de cet ouvrage se trouvera alors pleinement satisfaite, s'il vous ouvre des perspectives auxquelles vous n'auriez même pas songé, ... en trois dimensions bien entendu.



L'AUTEUR

Thomas Lachand-Robert est né en décembre 1966. De formation essentiellement scientifique, il prépare l'école Polytechnique.


0157 0286 148 F
9 782736 101572

SYBEX

AMSTRAD GRAPHISME EN TROIS DIMENSIONS



Document numérisé avec amour par

AMSTRAD CPC

MÉMOIRE ÉCRITE



https://acpc.me/